Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica

Un template per la PA: linee guida, accessibilità e CMS Umbraco

Tesi di laurea

Relatore
Prof.Ombretta Gaggi

Laureando
Matteo Noro
1229145

Anno Accademico 2021-2022

"Ogni bambino è un artista. Il problema è come rimanere un artista quando si cresce."

— *Pablo Picasso*

Dedicato ai miei nonni Augusto e Osvaldo, entrambi a loro modo maestri; conscio che oggi sarebbero fieri dei miei traguardi.

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecento ore, dal laureando Matteo Noro presso l'azienda SCP srl. Scopo primario del tirocinio è stato quello di sviluppare un Template per il deploy rapido di soluzioni web per aziende ed enti della Pubblica Amministrazione. Il prodotto pone fondamenta sulle linee guida per i servizi digitali della Pubblica Amministrazione e prende vita sul CMS Umbraco, basato su framework ASP.NET Core, e l'utilizzo della libreria grafica Bootstrap Italia. Per la natura stessa del progetto, lo sviluppo è stato fatto in ottica "Accessible By Default", per offrire massimi standard qualitativi congiuntamente a funzionalità e design.

"Créer, c'est vivre deux fois."

— Albert Camus

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine alla Professoressa Ombretta Gaggi, relatrice della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

Desidero ringraziare con affetto i miei genitori e miei fratelli per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio. Ogni opportunità non sarebbe stata tale senza di loro.

Ho desiderio di ringraziare poi i miei amici, compagni di corso e non, per l'aiuto e gli anni trascorsi assieme. Grazie a loro ho scoperto la forza nella collaborazione e nella condivisione di idee.

Ringrazio ancora Marika, per avermi sempre sostenuto ed essermi stata vicina durante questi anni di studio. Grazie a lei ho potuto ritrovare sempre la serenità necessaria ad affrontare ogni difficoltà, e il sentimento che dà forza ad ogni obiettivo.

Elevo infine un ringraziamento ai miei nonni, per i valori che mi hanno insegnato e la loro incrollabile fiducia nel mio operato.

Padova, Settembre 2022 Matteo Noro

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 L'idea

Ad oggi i siti web per la Pubblica Amministrazione presentano una forte frammentazione in termini di gestione e tecnologie utilizzate. Pur dovendosi basare su linee guida ben definite, spesso si trovano siti web affini che forniscono l'informazione, i contenuti, in modo incoerente sia rispetto alle normative sia all'interno dello stesso ecosistema. Inoltre, gli strumenti utilizzati appesantiscono molto il flusso di lavoro e il carico sul browser, rendendo negativa l'esperienza finale dell'utente/cittadino. Un altro grande problema è l'accessibilità: dato il fine che si pongono questi prodotti, l'accessibilità non deve essere una possibilità, ma una necessità, caratteristica spesso del tutto inesistente in questi servizi che, per loro stessa natura, dovrebbero permettere l'inclusività di qualsiasi categoria di utente.
L'obiettivo di questo progetto è quindi quello di creare un *Template* unico utile a tutte le categorie di servizi della Pubblica Amministrazione che vi si possono adeguare; garantendo massima flessibilità di gestione dei contenuti e offrendo, allo stesso tempo, una struttura ben definita e coesa come descritta nelle linee guida di design per i servizi della Pubblica Amministrazione[1].
A questo fine si utilizza il *CMS*[g] (*Content Management System*) *Umbraco*[g][2]: una piattaforma leader del settore che permette di creare da zero un dominio web, integrando un potente e altamente personalizzabile *Backoffice*[g] per la gestione dei contenuti. Tale strumento permette quindi di sviluppare un prodotto adattato alle proprie esigenze e, in questo caso, "*Accessible By Default*".
Per la resa grafica del prodotto, invece, viene utilizzata la libreria *Bootstrap Italia*[3], un tema basato su *Bootstrap*[g] che mette a disposizione una ampia gamma di componenti stilistiche per velocizzare lo sviluppo e adattarlo agli standard così per come pensati da *Designers Italia*.
La combinazione di questi elementi e un corretto approccio orientato all'*Human Centered Design* permette di ottenere un prodotto accessibile, veloce, ben strutturato e fruibile da qualsiasi categoria di dispositivo; offrendo inoltre al gestore finale del sito, manutentore della piattaforma, un facile e potente strumento di gestione dei contenuti.
Questa tipologia di prodotto viene inoltre incentivata dal bando "Esperienza del

[1] *Linee guida di design per i servizi digitali della PA*. URL: https://shorturl.at/efvIS.
[2] *Umbraco CMS*. URL: https://umbraco.com.
[3] *Bootstrap Italia*. URL: https://bootstrap-italia-next-development.vercel.app/.

Cittadino nei servizi pubblici[4]" che stanzia un fondo di 400 milioni di euro per la realizzazione di piattaforme da offrire a Comuni e enti della Pubblica Amministrazione. Per supportare lo sviluppo del *Template*, questo verrà inizializzato e perciò utilizzato su un caso concreto, cioè per la realizzazione del sito web per la residenza anziani di Auronzo Di Cadore, chiarendo che anche le case di riposo e le RSA[g]sono enti della PA[g]. Il prodotto sarà quindi costruito sulla base di questo esempio concreto, comprensivo di tutte le caratteristiche necessarie all'adattamento, poi, ad ogni genere di sito web per la Pubblica Amministrazione.

1.2 L'azienda

La Via al Digitale, dal Veneto lungo il Nord Italia.

L'azienda ospitante è la *SCP srl*[5], azienda che opera, nel Bellunese e nel Trevigiano, da più di 40 anni.
Propongono integrazione di applicativi esistenti e soluzioni digitali che migliorano i processi informativi e organizzativi di Aziende, Studi Professionali e Amministrazioni Pubbliche.
SCP è inoltre partner di altre Aziende di livello come SISTEMI Città per Treviso e Belluno, DedaNext e Sanmarco Informatica. Offre una vasta gamma di servizi, fra cui:

* Soluzioni per aziende e imprese: soluzioni per la *digital innovation* 4.0, consulenza, servizi e software per la gestione aziendale;
* Soluzioni per professionisti: 4.0, studio digitale e consulenza;
* Soluzioni per la PA digitale: formazione, infrastrutture, software, web e consulenza.

Attualmente la *SCP srl* dispone di oltre 50 risorse professionali di provenienza tecnico-applicativa con esperienza in gestione e creazione di progetti informatici complessi, nonché di numerose le certificazioni di prestigio consolidate da anni fra cui:

* Business Partner IBM dal 1983;
* Solution Partner Sistemi dal 1991;
* Microsoft Certified Partner dal 1997;
* Cisco Solutions Network dal 2002;
* Citrix Certified Network Associate dal 2002;
* Preferred Partner HP dal 2006;
* Sirv-Interop Certified Producer e Sirv-Interop Compliant Service dal 2007;
* Certificazione Aziendale di Qualità UNI EN ISO 9001:2015[6] (dal 1998);
* Certificazione alla Formazione EA 33-37 dal 2011.

[4] *Bando "Esperienza del Cittadino nei servizi pubblici"*. URL: https://shorturl.at/kGMQW.
[5] *SCP srl*. URL: https://www.scponline.it/.
[6] *Certificazione Aziendale di Qualità UNI EN ISO 9001:2015*. URL: https://www.scponline.it/media/1775/2021certificato-iso9001-2015.pdf.

1.3 Obiettivi

Notazione

Si farà riferimento ai requisiti secondo le seguenti notazioni:

* *O* per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
* *D* per i requisiti desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
* *F* per i requisiti facoltativi, rappresentanti valore aggiunto non strettamente competitivo.

Le sigle precedentemente indicate saranno seguite da una coppia sequenziale di numeri, identificativo del requisito.

Obiettivi fissati

Si prevede lo svolgimento dei seguenti obiettivi:

* Obbligatori
  - *O01*: Sviluppo del template;
  - *O02*: Proficuo lavoro in team;
  - *O03*: Realizzazione in autonomia di almeno una parte della soluzione;
* Desiderabili
  - *D01*: Completamento della soluzione con ampio contributo da parte dello studente;
* Facoltativi
  - *F01*: Interfacciamento con gestionale in uso preso l'ente pubblico;
  - *F02*: Realizzazione di un package Umbraco per un rapido deploy della soluzione.

1.4 Organizzazione del testo

Il secondo capitolo fornisce una infarinatura generale e riassunta delle linee guida di design per i servizi digitali della PA;

Il terzo capitolo descrive l'analisi dei requisiti, comprensiva quindi dei casi d'uso e il tracciamento dei requisiti;

Il quarto capitolo approfondisce gli strumenti e le tecnologie utilizzate;

Il quinto capitolo approfondisce il processo di sviluppo analizzando le varie parti che compongono il sito cui è stato applicato il *Template*;

Il sesto capitolo tratta i test che hanno accompagnato l'intero processo di sviluppo per verificarne correttezza e accessibilità;

Nel settimo capitolo si traggono le conclusioni sull'esperienza di tirocinio.

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

* gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
* per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: parola[g];
* i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*;
* Il prodotto finale viene riferito anche con la dicitura "Template", cioè con l'iniziale maiuscola; questo per evitare ambiguità con i template[g]del CMS.

Capitolo 2

Fondamenti linee guida della PA

Questo capitolo tratta i fondamenti sulle linee guida dello sviluppo di servizi web per la Pubblica Amministrazione

2.1 Human centered design

L'**utente al centro di tutto**. Qualsiasi servizio per la Pubblica amministrazione deve essere costruito e basare il suo design sull'utente, considerandone quindi le effettive esigenze come punto di partenza per pensare, costruire e migliorare i servizi digitali. Questo concetto copre tutte le fasi di ideazione: per una corretta allocazione delle risorse, basata sull'identificazione delle priorità e l'adozione di standard che evitano sprechi e duplicazioni di attività; per la realizzazione di servizi digitali efficaci, moderni, che facciano risparmiare tempo e inutili complicazioni agli utenti.

Alla base di questo principio vi sono vari obiettivi e conseguenti azioni chiave, fra cui:

* Focalizzarsi sulle priorità e tradurre gli obiettivi in indicatori misurabili;
* Chiarire e rappresentare le future funzionalità del servizio tramite user story;
* Miglioramento continuo;
* Accessibilità;
* Inclusività;
* Permettere il raggiungimento del sito da parte dei motori di ricerca e il raggiungimento semplice dei contenuti interessati;
* Costruire interfacce semplici da usare, anche su mobile;
* Utilizzare soluzioni comuni per ridurre costi e per aumentare l'efficacia;
* Gestire i dati nel rispetto della privacy e del GDPR.

2.2 Service Design

Per **Service Design** si intende quell'approccio, durante la fase di progettazione, con il fine di migliorare le caratteristiche di un servizio, orientando funzionalità, processi e componenti intorno alla effettive esigenze degli utenti. Si occupa quindi di definire in modo ottimale come si dovrebbe svolgere la relazione tra un utente e un'organizzazione. Quando l'organizzazione si identifica nella **Pubblica Amministrazione**, l'utente è il **cittadino**: l'interazione avviene tramite una serie di canali chiamati *touchpoint*, quali definiscono le possibilità di relazione tra le due parti.
In questo modo si forniscono al cittadino degli strumenti per svolgere attività specifiche e raggiungere i propri obiettivi, e dall'altro lato alla Pubblica Amministrazione un modo per rendere disponibili i propri servizi. Il Service Design si sintetizza pertanto nei seguenti punti:

* Partire dai bisogni dei cittadini tramite strumenti di analisi e attività, quindi, di user research;
* Trasparenza e collaborazione, tramite documentazione;
* La progettazione dei servizi deve utilizzare al meglio metodologie, tecnologie, componenti standard indicate nel Piano per l'informatica nella PA;
* Il principio della UE "digitale by default", secondo cui il servizio deve essere organizzato in forma digitale, e a partire da questo bisogna progettare altri punti di contatto con il cittadino in modo da abbracciare un'ottica multicanale, che consideri in modo integrato ogni modalità di erogazione del servizio, digitale e fisico.
* Semplificare;
* Misurare i risultati e tendere al miglioramento continuo.

2.3 Gestione dei progetti

2.3.1 Project Management

Nel ciclo di vita di un progetto è necessario identificare le seguenti figure e strumenti:

1. **Product Owner**;
2. **Product Manager**;
3. **Strumenti e ambienti per la gestione del progetto**.

2.3.2 Tipologie di progetti

Ci sono varie tipologie di progetto:

* **Ottimizzazione di servizi esistenti**: è necessario raccogliere dati sul loro utilizzo tramite *web analytics*, interviste utente oppure *usability test*. Sulla base di questi elementi sarà possibile mappare l'attuale esperienza utente dei diversi profili coinvolti (*user journey*), evidenziare le criticità e immaginare quali percorsi sia necessario migliorare (*user story*).

* **Riprogettazione di servizi esistenti in chiave digitale**: adottare una prospettiva più ampia in fase iniziale: capire le necessità degli utenti coinvolti (*persona*), l'intero sistema che supporta l'erogazione del servizio e verificare quali aspetti possano essere digitalizzati e quali no. Successivamente si potrà passare alla progettazione, sempre tramite *user story* con continui test di usabilità.

* **Creazione di nuovi servizi**: necessita uno sguardo ancora più ampio, partendo dalla mappatura di tutti gli stakeholder coinvolti e delle loro reciproche relazioni. Sarà necessario raccogliere il punto di vista degli utenti tramite attività di ricerca sul campo (intervista in contesto e osservazione), per capire al meglio le loro attuali criticità e necessità. I risultati delle fase di ideazione possono essere a loro volta formalizzati in una serie di proposte di design (*information architecture, flussi di esperienza e storie*), da prototipare e validare prima di procedere all'esecuzione finale del progetto.

2.3.3 Conoscere gli utenti

Avere un'idea chiara delle necessità delle persone che utilizzano i servizi che progettiamo, e conoscere nel dettaglio la loro esperienza di interazione con i canali digitali o fisici che rappresentano il servizio, è fondamentale per costruire una base solida su cui strutturare il progetto o da cui partire per migliorarlo.
Esistono in particolare due strumenti chiave che facilitano la compressione degli utenti:

* *persona*: o profili utenti; rappresentazioni astratte degli utenti che aiutano il team ad analizzare i loro bisogni e immaginare soluzioni concrete che rispondano ai loro problemi;

* *user story*: o mappature dell'esperienza; come metodo di analisi e progettazione dell'interazione con il servizio. La mappa dell'esperienza viene costruita mettendo sull'asse orizzontale tutte le fasi in cui l'utente interagisce con il servizio seguendo una logica temporale. Per ogni fase vengono quindi elencate le attività e i touchpoint con cui l'utente interagisce, costruendo una rappresentazione sintetica dell'esperienza.

2.4 Accessibilità

2.4.1 Definizione

Per accessibilità si intende la capacità dei sistemi informatici di erogare servizi e fornire informazioni fruibili, senza discriminazioni, anche a coloro che a causa di disabilità necessitano di tecnologie assistive o configurazioni particolari.
Nessun utente deve essere discriminato e deve quindi poter accedere alle informazioni e ai servizi digitali erogati dalla Pubblica Amministrazione.

2.4.2 Principi

L'accessibilità è basata su quattro solidi principi:

* Percepibile: le informazioni e i componenti dell'interfaccia utente devono essere presentati agli utenti in modi in cui essi possano percepirli. Questo è possibile tramite alternative testuali a immagini e media in generale, audiodescrizioni e sottotitoli;

* Utilizzabile: i componenti e la navigazione dell'interfaccia utente devono essere utilizzabili anche da tastiera senza impedimento alcuno. Bisogna quindi aver cura di disporre l'informazione secondo uno schema logico preciso e ordinato, permettendo una fruizione consequenziale del contenuto tramite input differenti dal mouse;

* Comprensibile: le informazioni e le operazioni dell'interfaccia utente devono essere comprensibili, con un linguaggio semplice, senza ambiguità e con etichette o suggerimenti sulla natura di un componente. A questo fine si deve lavorare sul linguaggio utilizzato, rendendolo uniforme alla lingua dichiarata e indicando le parole straniere, sulle abbreviazioni, sulle etichette, gli input e le istruzioni;

* Robusto: il contenuto deve essere abbastanza solido per essere interpretato in maniera affidabile da una grande varietà di programmi utente, comprese le tecnologie assistive. Importante quindi è anche saper fornire la massima compatibilità con i programmi attuali e futuri.

Le PA hanno l'obbligo di adattarsi alle WCAG 2.1 e devono rilasciare una **dichiarazione di accessibilità**.
Ogni sezione delle Linee Guida è costruita in ottica di accessibilità in ogni suo aspetto e integrazione.

2.5 Normative

La normativa italiana obbliga le PA a pubblicare determinate informazioni nei loro siti web. Fra queste, i contenuti minimi obbligatori sono:

* Amministrazione Trasparente;

* Pubblicità legale;

* Partita IVA;

* PEC;

* Pubblicazione atti di carattere normativo e amministrativo generale;

* Trattamento dati personali.

Inoltre, ogni servizio della Pubblica Amministrazione deve adeguarsi al **GDPR**, secondo il regolamento (UE) 2016/679132 del Parlamento Europeo del Consiglio del 27 aprile 2016 relativo alla protezione delle persone fisiche con riguardo al trattamento dei dati personali, nonché alla libera circolazione di tali dati e che abroga la direttiva 95/46/CE (regolamento generale sulla protezione dei dati)

2.6 Prototyping

2.6.1 Dalla progettazione alla prototipazione

La progettazione di un ambiente digitale si basa sui risultati delle attività di user research e co-progettazione con gli utenti, usate per far emergere i bisogni effettivi delle persone per cui si sta progettando.

Un buon metodo di lavoro può essere la stesura di liste di **bisogni** ordinati per priorità, ai quali affiancare la relativa funzione da progettare per soddisfarli.
Tra gli strumenti a disposizione per affrontare questa fase, uno dei più utilizzati è quello delle *user story*, che permette una rappresentazione ordinata delle azioni che il sistema dovrà realizzare per rispondere ai bisogni dei diversi utenti coinvolti.
La fase successiva del processo di progettazione è la creazione di **prototipi** che consentano al gruppo di lavoro di esplorare rapidamente una o più soluzioni alternative. È questo lo scopo della prototipazione: utile a verificare e comunicare le principali funzioni d'uso di un prodotto e offrire un'idea dell'ambiente informativo in cui l'utente si troverà a interagire per raggiungere il proprio scopo. Questo attua la simulazione **dei casi d'uso**.

2.6.2 Stortyboard e scenari d'uso

Attraverso una sintesi dei dati di ricerca, gli **scenari d'uso** permettono di definire soluzioni progettuali che tengono al centro le *persona*.
Questi descrivono in modo realistico la sequenza di azioni che gli utenti compiono all'interno del servizio. Si tratta di una narrazione macroscopica, non troppo dettagliata: una sorta di sceneggiatura all'interno della quale, con un approccio più analitico, si possono generare le **user story**. All'interno di uno scenario possono esistere più user story, che specificano con maggior dettaglio un preciso caso d'uso del servizio.
Un esempio di scenario d'uso è il seguente:

iscrizione all'asilo nido	1. Arrivo sul sito dell'istituto scolastico 2. Individuo la sezione dedicata alle iscrizioni 3. Accedo al percorso di iscrizione con SPID 4. Inserisco tutte le informazioni richieste 5. Ricevo conferma dell'avvenuta iscrizione, e le indicazioni per contattare la scuola in cui ho iscritto mio figlio

Le **user story** sono una descrizione informale delle funzioni di un servizio, espressa dal punto di vista dell'utente secondo una struttura che definisce il ruolo di chi la esprime, l'azione che vuole o deve compiere e l'obiettivo che muove l'azione.
La "formula" che sintetizza una user story si può esprimere come segue:

Io come [persona] vorrei [funzione] per [bisogno]

Riprendendo l'esempio soprastante quindi, si può andare nel particolare con la seguente user story di esempio: "Io come genitore vorrei compilare on line il modulo per iscrivere mio figlio al nido".
Ai fini di questo progetto, sono stati realizzati dei più tradizionali *use case* che evidenziano ugualmente uno scenario principale d'uso da parte di un utente.

2.6.3 Prototipi

Un prototipo è un modello sperimentale che permette di testare un'idea in maniera rapida ed economica, permettendo al team di rifinire il progetto o di valutare cambiamenti di approccio. Nella prima fase il prototipo è **low-fi** (low fidelity), a bassa fedeltà. Questa fase si avvale di *Wireframes*, cioè un'illustrazione a due dimensioni dell'interfaccia di una pagina, con lo solo scopo di organizzare gli elementi interattivi e dei blocchi di contenuto nello spazio di schermo.
La prototipazione **hi-fi** (high fidelity) interviene in un secondo momento, quando

l'organizzazione semantica e i flussi d'interazione sono stati validati grazie al prototipo low-fi ed è possibile progredire nella progettazione delle schermate inserendo gli elementi d'interfaccia (vedasi figma, adobe XD, ecc.).

2.7 Architettura dell'informazione

2.7.1 Scopo

Una buona architettura dell'informazione aiuta le persone a comprendere ciò che le circonda e a trovare ciò che cercano, sia online che offline. Lavorare su questo ambito implica una riflessione sulla struttura dell'informazione e sul linguaggio. L'architettura dell'informazione è più efficace se è progettata intorno ai reali bisogni delle persone: per questo si parla di *User Centered design*.

2.7.2 Applicazione

Affinchè vi sia una buona architettura dell'informazione bisogna saper **mappare il contenuto**, cioè le informazioni di tipo non strutturato (testi, immagini, video) o strutturato (dati e metadati) veicolate da pagine web.

In fase di progettazione, i contenuti di un sito web devono essere organizzati in diverse tipologie o *Content Type*[g]. Esempi di content type sono una scheda di presentazione di un servizio, una form per inserire dati anagrafici, una notizia o una scheda di presentazione di un evento.

Un sito web presenta abitualmente un **sistema di navigazione principale**, che a sua volta può essere organizzato in uno o più livelli e che genera il menù di navigazione di un sito web. La struttura di navigazione può essere riprodotta anche attraverso la creazione di *breadcrumbs*.

Fondamentale è la **Home Page**, che ha la funzione di punto d'ingresso ed è tipicamente il luogo in cui l'utente ottiene una visione chiara della missione di un sito e delle sue funzioni chiave. Un modo semplice per organizzare la home page è definire una struttura coerente rispetto al sistema di navigazione principale, per esempio attraverso un layout a fasce.

I siti web che offrono servizi digitali ai cittadini mettono a disposizione, se necessaria, un'**area personale** dell'utente a cui si accede mediante credenziali di accesso (per esempio SPID) e che possiede un proprio sistema di navigazione contestuale. In termini generali, l'area personale serve a gestire l'interazione di un utente con il sistema. Un modo semplice per organizzare un'area personale è prevedere:

* un'area messaggi;
* un'area che mostra la lista delle procedure in corso dei servizi attivati;
* un'area destinata ad archiviare l'esito delle azioni compiute in passato (es. lista dei pagamenti, dei documenti ricevuti, delle iscrizioni fatte).

Deve essere inoltre presente un **motore di ricerca**: questo ha il compito di fornire liste di risultati corrispondenti alle ricerche formulate dall'utente cercando tra i testi del sito

e/o utilizzando i sistemi di classificazione (come ad esempio categorie e tag) del sistema.

Al fine di controllare l'ambiente e garantire l'interoperabilità semantica, devono essere concordati un'**ontologia** e un **vocabolario concordato**.

2.8 SEO

Con il termine *search engine optimization* (SEO) si intende un insieme di tecniche iterative applicabili al contenuto delle pagine web e alla struttura dei siti che hanno lo scopo di migliorare il posizionamento di un contenuto web nel ranking dei risultati dei motori di ricerca. I fattori di ottimizzazione vengono generalmente suddivisi in 2 categorie: *fattori on-page*, cioè eseguibili all'interno del sito, e fattori *off-page*, cioè eseguibili al di fuori del sito.

2.8.1 Fattori on Page

I fattori on Page sono i seguenti:

* Titolo;
* Description (i motori troncano le description di oltre 160 caratteri spazi inclusi);
* Keywords;
* Originalità del contenuto e aggiornamento di esso;
* Paragrafazione e paginazione; nel caso ci sia la necessità di suddividere il contenuto in più pagine, è consigliabile:
  - specificare quale sia la pagina principale di visualizzazione attraverso l'attributo rel="canonical";
  - utilizzare gli attributi *HTML*[g](*HyperText Markup Language*) `rel="next"` e `rel="prev"`, per specificare la relazione di consequenzialità fra URL
* Grassetto;
* Immagini con testo alternativo;
* Anchor Text dei link, evitando espressioni povere;
* Struttura logica dei contenuti; scegliendo preferibilmente una struttura gerarchica;
* URL delle pagine:
  - impostare le URL in modo che contengano parole salienti e pertinenti rispetto ai contenuti della pagina che ospitano;
  - utilizzare i trattini (-) anziché gli underscore (_) per la punteggiatura;
  - cercare di ridurre il più possibile la lunghezza delle URL;
  - valutare l'utilizzo del file robots.txt per bloccare la scansione da parte dei *crawler* dei motori di ricerca delle URL con parametri dinamici (referral, ordinamenti, calendari...)
* Duplicazione dei contenuti:

  - Da evitare; in ambito SEO per duplicati si intendono contenuti molto simili o identici nell'ambito dello stesso sito ma in URL differenti;
  - In casi come il layout di stampa o presenza di una tabella dinamica che genera viste dello stesso contenuto ma URL dinamiche diverse, la duplicazione è necessaria. In questi e altri casi è possibile inviare a Google l'informazione di quale sia la pagina "master", o "canonica" da prendere in considerazione per l'indicizzazione.

* Mappa del sito: in xml, ha lo scopo di elencare le pagine web di un sito per comunicare a Google e altri motori di ricerca l'organizzazione dei contenuti;
* File Robots.txt: per ottimizzare processi di scansione per i crawler. Un file robots.txt è un file di testo memorizzato nella directory principale del sito che ha la finalità di indicare ai crawler dei motori di ricerca quali parti del sito non sono accessibili e quindi controllare il traffico di scansione;
* Tempi di caricamento delle pagine;
* Le pagine AMP[g](*Accelerated Mobile Pages*) per i contenuti di tipo "news";
* Dati strutturati: personalizzare l'aspetto di un sito nella ricerca di Google o di altri motori di ricerca.

2.8.2 Fattori off Page

In ottica di ottimizzazione SEO di un sito, è necessario curare e monitorare iterativamente il processo di costruzione della rete di link che il sito riceve dall'esterno (*inbound links*).
I motori di ricerca valutano la natura, la provenienza e la qualità di tali link più che la loro quantità, considerandoli un elemento di autorevolezza del sito soprattutto se questi provengono da siti altrettanto autorevoli e se il loro processo di acquisizione è considerato spontaneo.
A tal fine v'è *Search Console*: una risorsa online offerta gratuitamente da Google che consente di monitorare, gestire e ottimizzare la presenza di un sito o di un'applicazione mobile nei risultati di ricerca.

2.9 Linguaggio e progettazione del contenuto

Il linguaggio è fondamentale per la corretta creazione di un contenuto, per poterne permettere la corretta fruizione da parte dell'utente.

2.9.1 Scrivere per le persone

È importante utilizzare un linguaggio semplice, lineare, chiaro e che tenga in primis conto del lettore. Prima di creare un contenuto, bisogna aver ben chiaro:

* chi sono gli utenti a cui ci si rivolge;
* qual è lo scopo della loro visita, ovvero qual è il bisogno a cui il contenuto deve rispondere.

Per individuare chi sono gli utenti target e quali siano i loro bisogni, si possono utilizzare strumenti di user research, come ad esempio:

* sessioni partecipative con gli utenti;
* i dati di web analytics;
* gli A/B test e i test di usabilità (sottosezione 2.10.1 e 2.10.2).

2.9.2 Progettare i contenuti

La fase di progettazione dei contenuti contribuisce a modellare l'ambiente cognitivo nel quale gli utenti si muoveranno alla ricerca di informazioni.
Il linguaggio dà forma all'ambiente, all'ecosistema in cui l'utente si muove e raccoglie informazioni. Un buon punto di partenza per definire le modalità di progettazione è tramite workshop dedicati al linguaggio e ai contenuti.
Le priorità da definire sono le seguenti:

* di cosa hanno bisogno le persone/gli utenti?
* quali sono i contenuti e le informazioni, da mettere in rilievo?
* che parole usano le persone per chiamare un servizio? Che nome dare dunque ai contenuti e ai servizi?

È utile, a tal fine, definire una checklist di contenuto per garantire la facilità di lettura.

2.9.3 Gestire i contenuti

Oltre alla definizione e alla progettazione di questi, i contenuti vanno naturalmente gestiti.
Questo significa tenerli sempre aggiornati e migliorare i propri contenuti per:

* rispondere in modo più efficace ai bisogni degli utenti;
* evitare refusi, errori o incongruenze;
* rispondere a nuovi bisogni informativi di cui non si era tenuto conto in precedenza;
* gestire i processi di pubblicazione ed evitare le duplicazioni.

L'attività di gestione richiede di avere un *content inventory*, la capacità di organizzare un processo di produzione di nuovi contenuti o di revisione di contenuti esistenti; nonché la eventuale migrazione di essi.
Bisogna porre inoltre attenzione alla proprietà intellettuale: la Pubblica Amministrazione suggerisce di adottare e indicare esplicitamente l'utilizzo della licenza *Creative Commons Linee guida PA 10 Attribution 4.0278 (codice SPDX: CC-BY-4.0)*, con la libertà di condividere e modificare.

2.10 User Research

La *User Research* risulta fondamentale per poter cogliere il grado di soddisfazione, nonché di *usabilità* del prodotto in base alle esigenze degli utenti, costruendo quindi il proprio progetto intorno al già citato *User Centered Design.*
Per *usabilità* si intende: *«il grado in cui un prodotto può essere usato da particolari utenti per raggiungere certi obiettivi con efficacia, efficienza, soddisfazione in uno specifico contesto d'uso» (ISO 9241-210:2010).*
Quest'ultima focalizza, quindi, la dimensione funzionale dell'interazione tra un sistema e l'utente in relazione a precisi obiettivi e contesti d'uso.

2.10.1 Test di usabilità

Si eseguono i test di usabilità secondo il protocollo **eGLU LG**.
La procedura di osservazione degli utenti, in breve, si svolge con le seguenti modalità:

* il conduttore dell'osservazione stila dei compiti da sottoporre ad alcuni partecipanti. I compiti, chiamati *task* dagli esperti, possono riguardare, per esempio, la ricerca di specifiche informazioni, la compilazione di moduli online, lo scaricamento di documenti;

* alcuni utenti vengono selezionati e invitati a partecipare;

* si chiede a ciascun utente di eseguire i *task* assegnati. Durante l'osservazione non si pongono domande dirette, ma si osservano le persone interagire col sito e le eventuali difficoltà che incontrano. I *task* possono essere eseguiti con successo o meno. Al termine dell'esecuzione si usano dei questionari per raccogliere informazioni sul gradimento e sulla facilità d'uso percepita;

* sulla base dei dati raccolti si può avere un'idea dei punti di forza del sito e delle sue criticità. Questo consente di apportare da subito modifiche in base ai problemi riscontrati, di approfondire le criticità con test avanzati condotti da esperti o di confrontare fra loro le criticità di versioni successive del medesimo prodotto.

2.10.2 A/B testing

L' **A/B Testing** è un metodologia di analisi che ha l'obiettivo di confrontare due versioni di una pagina web di un sito o di un'applicazione, che differiscono per un elemento specifico. Permette quindi di effettuare delle scelte di design basate sui dati. Gli utenti cui il test viene "somministrato" sono suddivisi in due gruppi ad ognuno dei quali viene mostrata una delle due diverse varianti/configurazioni.
Alla fine del test vengono analizzati e confrontati i dati derivati delle due versioni sperimentate; la variante con la performance migliore rispetto all'obiettivo del test verrà portata avanti nel percorso di sviluppo.

2.10.3 Ricerche qualitative e quantitative

La *User Research* ha come fine ultimo quello di studiare gli utenti per progettare servizi quanto più rispondenti alle loro effettive esigenze.
Questo obiettivo si raggiunge attraverso approcci di ricerca di tipo qualitativo e/o quantitativo.
Dove la **ricerca quantitativa** basa fondamenta su fini statistici, campioni numerosi e modelli matematici, la **ricerca qualitativa** cerca di comprendere le motivazioni sottese ad attitudini, comportamenti e atteggiamenti dell'utente, studiandone le attività, i contesti d'uso, le necessità ma anche gli errori e le frustrazioni.
Possiamo dividere la ricerca qualitativa in tre tipi:

* **ricerca esplorativa**: all'inizio di un progetto permette di analizzare un tema che non si conosce a fondo. Prevede interviste e osservazioni in contesto;

* **ricerca generativa**: coinvolgere gli utenti in sessioni di discussione e generazione di idee, utilizzando tecniche come il *focus group* e sessioni di co-design;

* **ricerca valutativa**: si svolge quando sono già disponibili i primi prototipi della soluzione progettata e si vuole raccogliere il feedback degli utenti nello sperimentare l'interazione con il servizio digitale in questione. Prevede test di usabilità e il *cognitive walkthrough*.

2.10.4 Web Analytics

A posteriori, vengono utilizzati anche strumenti di **Web Analytics** per raccogliere informazioni sull'utilizzo dei servizi.
Questi strumenti, il cui uso è da dichiarare nella *privacy policy*, sono utili per capire il modo in cui gli utenti interagiscono e fruiscono del prodotto, nonché per monitorarne il flusso di dati.
Un controllo adeguato e continuativo delle statistiche riportate contribuisce al miglioramento continuo del servizio, adattandosi alle esigenze e alle statistiche che possano render noti aspetti non prevedibili a priori.
Un'analisi sistematica dei dati statistici di performance e soddisfazione utente è fondamentale, quindi, per decidere quali azioni migliorative intraprendere in un servizio digitale.

2.11 User Interface

L'*interfaccia utente* è il ponte tra i servizi digitali e i loro destinatari. È tutto ciò con cui l'utente entra in relazione, nei vari contesti, per usufruire di un servizio o un prodotto digitale.
È quindi un concetto che va oltre al puro aspetto visivo, è il *touch point* del servizio digitale, è l'oggetto che muove l'utente all'interno dell'intero sistema.
Scopo primario dell'interfaccia di un servizio è quello di aiutare l'utente a raggiungere ciò che cerca in modo naturale ed immediato, quasi trasparente. Per questo la coerenza dei vari elementi che la compongono è fondamentale.
Bisogna inoltre avere particolare attenzione a inclusività e tolleranza agli errori: non ci si deve aspettare che l'utente abbia sempre chiaro ciò che vuole e sappia comprendere appieno eventuali istruzioni.
Compito del *designer* quindi è quello di progettare interfacce che sappiano accompagnare il cittadino nel percorso di ricerca, prevedendo diverse modalità d'utilizzo.

2.11.1 Modello di un'interfaccia

Il livello di dettaglio più basso viene definito attraverso la creazione di un modello (*wireframe*) dell'interfaccia utente.
Questa scelta assicura che l'attenzione sia incentrata sugli aspetti fondamentali della navigazione e della struttura, nel pieno rispetto dei requisiti di progetto e dei bisogni da soddisfare. Un prototipo *lo-fi* aiuta a organizzare gli elementi interattivi nello spazio dello schermo, definendo anche le modalità di interazione, comportamento e la sequenza dei passaggi (*workflow*) che un utente esegue per concludere un processo.
Il modello *hi-fi* si concentra invece a posteriori sui dettagli, le animazioni e lo stile. Adottando inoltre degli standard visuali, questo processo viene semplificato e favorisce la produzione successiva.

2.11.2 Caratteri

2.11.3 Tipografia

Un carattere che rispetti certi vincoli di leggibilità è fondamentale in termini di accessibilità per permettere a chiunque di fruire del servizio senza particolare disagi.
La famiglia di font usata per la PA è il *Titillium Web*: è stato scelto come *typeface* principale per i contenuti web grazie alla sua *x-height* ampia, alla struttura lineare e alla flessibilità d'uso.
Un *typeface* secondario è il *Roboto Mono*, introdotto nelle Linee Guida per la chiarezza e la leggibilità dei numeri, rendendolo pertanto adatto alla rappresentazione di calcoli matematici, numeri in tabelle e codice di programmazione.
Un terzo *typeface* (con grazie) è il *Lora*, nato espressamente per la lettura su display.

2.11.4 Dimensioni del carattere

La dimensione del corpo del testo, con riferimento ad esempio al font Titillium Web, non può essere inferiore a **16px** per uno schermo mobile e inferiore a **18px** per schermi grandi. Si possono utilizzare misure minori in caso di didascalie, note, o testi di secondaria importanza che per lunghezza o posizionamento nella pagina richiedano dimensioni ridotte.

2.11.5 Paragrafi

Per favorire una lettura confortevole del testo, questo non dovrebbe superare i **75 caratteri** per riga su desktop e non dovrebbe mai essere inferiore ai **15**. Bisogna utilizzare l'allineamento a sinistra, evitando allineamento giustificato senza sillabazione. I paragrafi possono essere distinti applicando uno spazio verticale fra di essi o, in alternativa, usando una indentatura di misura pari a quella dell'interlinea. L'interlinea (in inglese, *leading*), sia dei titoli che del corpo del testo, è calcolata tenendo conto di una immaginaria griglia di 8px, in modo da creare una sorta di "ritmo verticale" nella lettura.

2.11.6 Collegamenti

I link ad altre aree del servizio o a siti esterni devono avere un elemento di distinguibilità rispetto al testo normale. Pertanto è buona norma mantenere una sottolineatura specialmente se il link è inserito all'interno di un paragrafo. Alternativamente si può utilizzare il grassetto, come attualmente adottato nella libreria *Bootstrap Italia 2.0.0*.

2.11.7 Colore

Il colore è un elemento essenziale nella definizione di un'interfaccia: può servire a differenziare, connettere, evidenziare, nascondere. Contribuisce alla gerarchia visiva e può essere un elemento di supporto alla comunicazione. Il colore influisce fortemente sull'accessibilità del prodotto. Gli utenti affetti da disabilità visive come la deuteranopia, protanopia e tritanopia potrebbero non vedere bene i colori oppure non vederli affatto. È quindi necessario garantire un rapporto di contrasto minimo con lo sfondo di **4,5:1 (AA)** oppure, preferibilmente, di **7:1 (AAA)**.

2.11.8 Schema colore

La scelta dei colori è dettata dal materiale identitario dell'ente o agenzia (logo, stemma, gonfalone, ecc.). In uno schema colore distinguiamo il colore **base**, che viene utilizzato per una percentuale maggiore rispetto agli altri colori, i colori **secondari** e **neutri** (ad esempio grigio, bianco, ecc.).
Per i colori secondari si dovranno definire:

* colori strettamente connessi al colore base;

* un eventuale colore di risalto (chiamato *accent color*) utilizzato in misura minore poiché è associato a elementi che prevedono un'interazione, come bottoni, elementi di controllo (sliders, radio, ecc.), link, campi di testo.

La palette, però, deve essere costituita da non più di **5 tonalità**, dove non più di 3 avranno un differente valore di colore (*Hue*).

2.11.9 Griglie

All'interno dello spazio a disposizione, l'organizzazione del contenuto deve essere strutturata seguendo un sistema di **griglie responsive**, per mantenere un'efficace esperienza utente trasversalmente ai dispositivi utilizzati.
Le dimensioni delle colonne vanno adattate ai cambiamenti della *viewport*: ogni colonna occuperà una percentuale di spazio specifica a seconda che sia visualizzata su dispositivi desktop, tablet, o smartphone.
Il sistema di Grid viene automaticamente gestito dalle classi interne alla libreria *Boostrap Italia*.

2.11.10 Icone

Quando si utilizzano delle icone è necessario assicurare una chiara comprensione del loro significato. Pertanto ogni icona dovrà essere associata ad un *tooltip* o ad un piccolo testo che ne chiarisca l'azione.
La stessa icona non dovrà essere utilizzata per indicare azioni diverse all'interno della stesso contesto.
Un set di icone standard è anch'esso fornito e compreso nella libreria *Bootstrap Italia*.

2.11.11 Bottoni

Seguendo una gerarchia prestabilita, i bottoni si suddividono in base all'importanza e al ruolo, assumendo valori dal *Primary* fino al *Quaternary*.

2.11.12 Navigazione

I componenti che si possono inserire all'interno della navigazione sono molteplici. Nel kit di *Bootstrap Italia* ci sono diverse varianti di "Tab" utili a disporre i contenuti in base a diversi fattori di schermo, colore e stile.

2.11.13 Data display

Nella categoria *Data Display* sono compresi i componenti che hanno come funzionalità quella di mostrare informazioni in modo organizzato oppure evidenziato, come ad esempio gli "*Accordion*" o i "*Callout*".

2.11.14 Data entry

Esempi di componenti appartenenti alla categoria *Data entry* sono i campi di tipo "*Input*" che si utilizzano per costruire form. Il componente è costruito in modo da poter attivare o disattivare i diversi status: normale, avvertimento, errore e successo. L'etichetta del campo è indicativa di cosa va inserito.
(Nel caso di questo progetto, i form sono stati costruiti tramite un *Plugin* del CMS Umbraco).

2.11.15 Responsive Web Design

Il sito web deve sempre essere progettato e sviluppato con un approccio *responsive*, per garantire un'esperienza d'uso ottimale indipendentemente dalla risoluzione dello schermo.
Il *Responsive Web Design* si associa ad un ulteriore criterio di progettazione, volto a semplificare le interfacce: l'approccio *Mobile First*.

2.11.16 Mobile First

L'approccio **Mobile First** consiste nel valutare in prima istanza l'esperienza e le necessità per i dispositivi mobile, per poi arricchire di elementi e funzionalità la composizione della pagina mano a mano che la dimensione, le capacità computazionali e di rete del dispositivo aumentano. Si parte quindi dall'essenziale, da una interfaccia semplice, pulita e chiaramente interagibile. Viene quindi posta una "forzatura" nella progettazione, positiva a stabilire fin da subito dei criteri e delle scelte utili all'usabilità.
A questo concetto si lega inoltre quello di **Progressive enhancement** secondo cui la lavorazione inizia da un nucleo solido e irrinunciabile di contenuti che vengono arricchiti man mano che il dispositivo utilizzato dall'utente è più performante e all'avanguardia.
La **Graceful degradation** è inoltre un importante fattore secondo cui l'interfaccia deve rimanere navigabile e permettere comunque di accedere alle sue funzioni fondamentali man mano che viene fruita attraverso tecnologie meno moderne e interattive.

2.12 Feature detection

Per **Feature detection** si intende il **riconoscimento delle caratteristiche**. Il sito web può rilevare delle proprietà che caratterizzano il metodo di accesso da parte dell'utente.
Attraverso una rilevazione puntuale, è possibile sapere come indirizzare ogni aspetto dell'informazione che si vuole trasmettere.
Tali caratteristiche possono spaziare in termini di schermo utilizzato, dimensioni, risoluzione e densità di pixel, fino ai metodi di input.
A tal fine si utilizzano *Media-Queries*[g] e *Javascript*[g], che permette di analizzare qualsiasi funzionalità presente tra le Web API (*Application Programming Interface*), oltre a poter conoscere praticamente ogni dettaglio dell'utente collegato.

CSS[g] (*Cascading Style Sheets*) ha capacità ben più limitate, ma in modo simile alla regola *@media* si può utilizzare *@support* può verificare la corretta interpretazione di proprietà da parte dei browser su cui viene usata.

2.12.1 Supporto Browser

È necessario assicurare la compatibilità con versioni dei browser che abbiano una penetrazione media tra la popolazione di almeno 1 persona ogni 100 abitanti[1].
Con *Bootstrap Italia 2.0.0*[2] si garantisce il supporto per le seguenti versioni dei browser più utilizzati:

* Microsoft Edge > 16
* Firefox > 60
* Safari > 12 / iOS Safari > 12
* Chrome > 60

In linea con *Bootstrap 5*[3], il supporto a Internet Explorer 10 e 11 viene abbandonato.

2.13 Misurare le prestazioni

Vi sono due ambiti da differenziare: i **tempi di caricamento** e le **performance di esecuzione** della pagina.
Per i tempi di caricamento sono utili strumenti come *Google PageSpeed Insights*[4] e *WebPagetest*.
Questi strumenti evidenziano problemi come immagini esageratamente grandi o poco ottimizzate, oppure aiutano a calibrare altri fattori come sfruttare al meglio il *caching del browser* o dare priorità ai contenuti immediatamente visibili.
Per ottenere invece informazioni più dettagliate riguardo eventuali inefficienze di codice a runtime, si può far riferimento agli strumenti di analisi nei principali browser, come lo strumento per sviluppatori *LightHouse*[5] di Google Chrome o altri tool integrati in diverse soluzioni.

[1] *Browser Market Share Worldwide*. URL: https://gs.statcounter.com/.
[2] *Bootstrap Italia Next*. URL: https://github.com/italia/bootstrap-italia-next.
[3] *Bootstrap 5 Support*. URL: https://getbootstrap.com/docs/5.0/getting-started/browsers-devices/.
[4] *Google PageSpeed Insights*. URL: https://pagespeed.web.dev/.
[5] *Google Lighthouse*. URL: https://developer.chrome.com/docs/lighthouse/overview/.

Capitolo 3

Analisi dei requisiti

Questo capitolo riporta le persona, i casi d'uso identificati e conseguente tracciamento dei requisiti.

3.1 Confronto con gli stakeholder e linee guida

Gli obiettivi e i requisiti principali sono stati individuati a seguito di un confronto con gli *stakeholder* e basandosi principalmente sullo studio individuale delle linee guida per la Pubblica Amministrazione. Proprio quest'ultime definiscono un approccio standard alla progettazione dei servizi web, pertanto ne sono stati ripresi concetti e contenuti per rielaborarli ai fini del progetto.

Si precisa che il *Template*, per sua stessa natura, è riadattabile a svariati contesti, che richiedono quindi requisiti differenti. In questo caso, si analizzeranno le *persona*, i requisiti e i casi d'uso che riguardano strettamente il sito per la residenza anziani. Naturalmente ogni sito che si andrà a creare sul *Template* avrà variabili differenti, applicabili tramite la manipolazione libera dei contenuti offerta nel *Backoffice*.

3.2 Personas

Il primo passo è individuare delle *persona*, ovvero rappresentazioni astratte degli utenti che aiutano il team ad analizzare i loro bisogni e immaginare soluzioni concrete che rispondano ai loro problemi. Sono stati individuati quindi i seguenti profili:

* Persona di media età che accede per la prima volta al sito;
* Persona di media età che entra abitualmente nel sito;
* Persona di media età che vuole approfondire nel dettaglio la struttura;
* Persona di media età che deve recuperare velocemente una informazione sulla struttura;
* Persona anziana che vuole effettuare una segnalazione e vedere le ultime notizie;

3.3 Entità

Individuato lo stereotipo di utente, si sono definite di seguito le entità presenti nell'intero ecosistema del sito. Sono state individuate dunque le seguenti:

Sito Web

Il servizio in sè, offerto al cittadino/utente e gestito internamente dalla figura dell'amministratore.

Sezione

Sezione del sito web, che contiene una porzione di contenuti all'interno del dominio. Una sezione può essere:

* Pagina diretta, senza un nodo padre (ad eccezione della HomePage);
* Coppia di pagine *Container*-Singola pagina, ovvero i contenuti principali hanno un nodo padre che li contiene e li raggruppa in base a dati specifici, presentandone un'anteprima rapida.

Cittadino/Utente

Un utente che accede al sito web. Questo può essere alla sua prima visita o meno, e deve poter recepire le informazioni che offre il sito senza riscontare difficoltà nell'uso. Attualmente non è prevista alcuna forma di autenticazione.

Amministratore dei contenuti

Amministratore del sito web, a cui viene offerto il *Backoffice* di Umbraco per la pubblicazione e la modifica dei contenuti. Tutti i dati di cui bisogna tener traccia vengono automaticamente gestiti e salvati dal CMS.

3.4 Navigazione nel sito da parte dell'utente

Un utente, o cittadino, appena entrato nel sito deve in primo luogo poter navigare in esso attraverso le sezioni messe in evidenza nell'Header. Deve poter inoltre vedere le ultime notizie pubblicate internamente al sito, e avere un collegamento rapido alle sezioni di amministrazione trasparente e albo pretorio. In caso di necessità deve poter inviare una segnalazione alla struttura nonché, a sua discrezione, una valutazione del sito. Se l'utente vuole accedere velocemente ad una contenuto specifico, deve poter cercare una parola chiave nella ricerca globale offerta dal sito. Se all'interno di un contenuto si presentano degli allegati scaricabili, deve poter effettuare il download di questi. Infine, se una pagina si riferisce ad un argomento, una tipologia o una categoria, l'utente deve poter visualizzare tutte le pagine correlate.

UC1.1 Navigare nelle sezioni principali

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino è entrato nel sito web;

* **Postcondizioni:** L'utente/cittadino è entrato in una sezione del sito di livello inferiore o ha aperto la finestra di un menu *dropdown*;
* **Scenario principale**
  1. L'utente/cittadino entra nel sito web;
  2. L'utente/cittadino preme su una voce del menu principale nell'header;
  3. L'utente/cittadino è entrato in una sezione del sito di livello inferiore o ha aperto un menu *dropdown*.

UC1.1.1 Visualizzare una notizia

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino è entrato nella sezione novità;
* **Postcondizioni:** L'utente/cittadino visualizza l'intera notizia;
* **Scenario principale**
  1. L'utente/cittadino entra nella sezione novità;
  2. L'utente/cittadino preme su una notizia presentata nella pagina di novità;
  3. L'utente/cittadino vede la notizia completa su una nuova finestra.

UC1.1.2 Visualizzare informazioni sulla struttura *

Questo UC, come altri segnati da un "", offre un caso nel dettaglio dell'use case generico UC1.1.x.*

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino preme sulla voce "Chi siamo" nell'header e apre la finestra con la lista di link;
* **Postcondizioni:** L'utente/cittadino visualizza una pagina con informazioni sulla struttura;
* **Scenario principale**
  1. L'utente/cittadino preme sulla voce "Chi siamo" nell'header;
  2. L'utente/cittadino preme su un link di interesse nella lista apparsa nella tendina;
  3. L'utente/cittadino vede la pagina con l'informazione completa sull'argomento.

UC1.1.3 Visualizzare tariffe e modulistica *

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino preme sulla voce "Tariffa e modulistica" nell'header e apre la finestra con la lista di link;
* **Postcondizioni:** L'utente/cittadino visualizza una pagina con informazioni sulle tariffe e la modulistica della struttura;

* **Scenario principale**
    1. L'utente/cittadino preme sulla voce "Tariffa e modulistica" nell'header;
    2. L'utente/cittadino preme su un link di interesse nella lista apparsa nella tendina;
    3. L'utente/cittadino vede la pagina con l'informazione completa sull'argomento.

UC1.1.4 Visualizzare orari e contatti

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino preme sulla voce "Orari e contatti" nell'header;
* **Postcondizioni:** L'utente/cittadino visualizza una pagina con gli orari delle visite o i contatti della struttura;
* **Scenario principale**
    1. L'utente/cittadino preme sulla voce "Orari e contatti" nell'header;
    2. L'utente/cittadino preme una sezione di interesse di orari o contatti;
    3. L'utente/cittadino vede la pagina con l'informazione completa sugli orari o i contatti.

UC1.2 Visualizzare form di segnalazione

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino preme sulla voce "Segnalazioni" nell'header;
* **Postcondizioni:** L'utente/cittadino visualizza una pagina con un form di segnalazione;
* **Scenario principale**
    1. L'utente/cittadino preme sulla voce "Segnalazioni" nell'header;
    2. L'utente/cittadino visualizza il form di segnalazione da compilare.

UC1.2.1 Inviare una segnalazione

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino è entrato nella pagina di segnalazioni;
* **Postcondizioni:** L'utente/cittadino invia una segnalazione;
* **Scenario principale**
    1. L'utente/cittadino compila i campi del form;
    2. L'utente/cittadino preme sul tasto "invia";
    3. L'utente/cittadino invia correttamente la segnalazione e riceve un messaggio di avvenuta operazione.

* **Scenario secondario**

    1. L'utente/cittadino compila i campi del form in modo errato o incompleto;
    2. L'utente/cittadino riceve dei messaggi di errore nei campi sbagliati.

UC1.3 Aprire la ricerca

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino preme sul pulsante di ricerca;
* **Postcondizioni:** L'utente/cittadino apre la modale per la ricerca;
* **Scenario principale**

    1. L'utente/cittadino preme sul tasto di ricerca;
    2. Si apre la modale con il campo per la ricerca.

UC1.3.1 Effettuare la ricerca

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino ha aperto la modale per la ricerca;
* **Postcondizioni:** L'utente/cittadino visualizza i risultati della ricerca;
* **Scenario principale**

    1. L'utente/cittadino inserisce il termine o la frase da ricercare;
    2. Si apre una nuova finestra con i risultati della ricerca.

* **Scenario secondario**

    1. L'utente/cittadino inserisce un input vuoto o di un termine non presente nel sito;
    2. Si apre una nuova finestra con un messaggio che avvisa della fallita ricerca e un nuovo campo di input.

UC1.4 Visualizzare la pagina di amministrazione trasparente

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino preme sulla voce "Amministrazione trasparente" nella HomePage;
* **Postcondizioni:** L'utente/cittadino viene reindirizzato ad una pagina con l'amministrazione trasparente;
* **Scenario principale**

    1. L'utente/cittadino preme sulla voce "Amministrazione trasparente" nella HomePage;
    2. si apre una nuova finestra con la pagina di amministrazione trasparente.

UC1.5 Visualizzare la pagina dell'albo pretorio

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino preme sulla voce "Albo pretorio" nella HomePage;
* **Postcondizioni:** L'utente/cittadino viene reindirizzato ad una pagina con l'albo pretorio;
* **Scenario principale**
  1. L'utente/cittadino preme sulla voce "Albo pretorio" nella HomePage;
  2. Si apre una nuova finestra con la pagina dell'albo pretorio.

UC1.6 Visualizzare i luoghi di interesse

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino preme sulla voce "Intorno a noi" nella HomePage;
* **Postcondizioni:** L'utente/cittadino visualizza una pagina con la lista dei luoghi di interesse;
* **Scenario principale**
  1. L'utente/cittadino preme sulla voce "Luoghi di interesse" nella HomePage;
  2. L'utente/cittadino visualizza la lista di tutti i luoghi di interesse in una nuova finestra.

UC1.6.1 Visualizzare un luogo

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino si trova nella pagina dei luoghi di interesse;
* **Postcondizioni:** L'utente/cittadino visualizza una pagina con i dettagli del luogo;
* **Scenario principale**
  1. L'utente/cittadino entra nella sezione dei luoghi di interesse;
  2. L'utente/cittadino preme sulla voce di un luogo;
  3. L'utente/cittadino visualizza i dettagli del luogo in una nuova finestra.

UC1.7 Visualizzare form di valutazione del sito

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino preme sulla voce "Valuta questo sito" nel Footer;
* **Postcondizioni:** L'utente/cittadino visualizza una pagina con un form per la valutazione del sito;

* **Scenario principale**
    1. L'utente/cittadino preme sulla voce "Valuta questo sito" nel Footer;
    2. L'utente/cittadino visualizza il form di valutazione da compilare.

UC1.7.1 Inviare una valutazione

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino è entrato nella pagina di valutazione del sito;
* **Postcondizioni:** L'utente/cittadino invia una valutazione;
* **Scenario principale**
    1. L'utente/cittadino compila i campi del form;
    2. L'utente/cittadino preme sul tasto "invia";
    3. L'utente/cittadino invia correttamente la valutazione e riceve un messaggio di avvenuta operazione.
* **Scenario secondario**
    1. L'utente/cittadino compila i campi del form in modo errato o incompleto;
    2. L'utente/cittadino riceve dei messaggi di errore nei campi sbagliati.

UC1.8 Visualizzare la scheda di una persona

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino è in una pagina con un riferimento ad una persona;
* **Postcondizioni:** L'utente/cittadino visualizza la pagina con i dettagli su quella persona;
* **Scenario principale**
    1. L'utente/cittadino preme sul riferimento alla persona;
    2. L'utente/cittadino vede i dettagli della persona in una nuova finestra.

UC1.9 Visualizzare le pagine correlate ad un argomento

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino è in una pagina con un riferimento a dato argomento;
* **Postcondizioni:** L'utente/cittadino visualizza la pagina con i tutti i contenuti che si riferiscono a quell'argomento;
* **Scenario principale**
    1. L'utente/cittadino preme sul riferimento all'argomento;
    2. L'utente/cittadino visualizza tutti i contenuti che si riferiscono a quell'argomento in una nuova finestra.

UC1.10 Visualizzare le pagine correlate ad una tipologia

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino è in una pagina con un riferimento ad una data tipologia;
* **Postcondizioni:** L'utente/cittadino visualizza la pagina con i tutti i contenuti che si riferiscono a quella tipologia;
* **Scenario principale**
    1. L'utente/cittadino preme sul riferimento alla tipologia;
    2. L'utente/cittadino visualizza tutti i contenuti che si riferiscono a quella tipologia in una nuova finestra.

UC1.11 Visualizzare le pagine correlate ad una categoria

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino è in una pagina con un riferimento ad una data categoria;
* **Postcondizioni:** L'utente/cittadino visualizza la pagina con i tutti i contenuti che si riferiscono a quella categoria;
* **Scenario principale**
    1. L'utente/cittadino preme sul riferimento a quella categoria;
    2. L'utente/cittadino visualizza tutti i contenuti che si riferiscono a quella categoria in una nuova finestra.

UC1.12 Scaricare un file allegato

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino è in una pagina con un allegato scaricabile;
* **Postcondizioni:** L'utente/cittadino scarica l'allegato;
* **Scenario principale**
    1. L'utente/cittadino preme sul riferimento all'allegato;
    2. Il browser scarica il file allegato nella macchina dell'utente/cittadino.
* **Scenario secondario**
    1. L'utente/cittadino preme sul riferimento all'allegato;
    2. Il browser va in errore e non scarica la risorsa.

UC1.13 Navigare fra gli antenati di una pagina

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino è in una pagina qualsiasi;
* **Postcondizioni:** L'utente/cittadino si muove in una pagina antenata di quella precedente;
* **Scenario principale**
  1. L'utente/cittadino preme sul link di un nodo antenato di quello attuale;
  2. L'utente/cittadino viene reindirizzato in una nuova pagina antenata di quella precedente e la visualizza correttamente.

Ulteriori use case non legati al sito della struttura anziani

UC1.1.5 Visualizzare un ufficio

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino è entrato nella sezione di amministrazione;
* **Postcondizioni:** L'utente/cittadino visualizza i dettagli dell'ufficio;
* **Scenario principale**
  1. L'utente/cittadino entra nella sezione di amministrazione;
  2. L'utente/cittadino preme sulla scheda di un ufficio presentata nella pagina di amministrazione;
  3. L'utente/cittadino vede tutti i dettagli dell'ufficio.

UC1.1.6 Visualizzare un servizio

* **Attori primari:** Utente/Cittadino;
* **Precondizioni:** L'utente/cittadino è entrato nella sezione dsei servizi;
* **Postcondizioni:** L'utente/cittadino visualizza i dettagli del servizio;
* **Scenario principale**
  1. L'utente/cittadino entra nella sezione dei servizi;
  2. L'utente/cittadino preme sulla scheda di un servizio presentata nella pagina dei servizi;
  3. L'utente/cittadino vede tutti i dettagli del servizio.

UC1.1.x Visualizzare pagina di informativa standard

Use case generico per tutte le pagine di tipo standard, descritte precedentemente nello specifico e segnate da un "".*

- **Attori primari:** Utente/Cittadino;
- **Precondizioni:** L'utente/cittadino preme sulla voce della sezione standard nell'header e visualizza o una nuova pagina *Container* o una finestra con la lista di link;
- **Postcondizioni:** L'utente/cittadino visualizza una pagina con le informazioni contenute nella pagina standard;
- **Scenario principale**
  1. L'utente/cittadino preme sulla voce della sezione standard nell'header;
  2. L'utente/cittadino viene reindirizzato ad una nuova pagina o preme su un link di interesse nella lista apparsa nella tendina;
  3. Se si trova in una nuova pagina, selezione il link di interesse;
  4. L'utente/cittadino vede la pagina con l'informazione completa.

3.5 Gestione del Backoffice da parte dell'amministratore

L'utente a cui viene affidata la amministrazione finale del sito è l'amministratore, il quale è responsabile, in primo luogo, di compilare e aggiornare i dati principali della struttura. Per eseguire qualsiasi modifica, deve naturalmente aver effettuato l'autenticazione nel *Backoffice* di Umbraco. L'amministratore deve poter pubblicare e/o modificare nuove pagine di contenuto di tipo: notizia, luogo, orari e contatti, argomenti, tipologia, categoria, di informativa standard e, non nel caso della residenza per anziani, pagine di uffici e servizi. Deve inoltre poter aggiornare i dati e i contenuti presentanti nella HomePage, nell'Header e nel Footer. Se necessario può decidere di mostrare un avviso importante nella zona *Above the fold* della HomePage.

UC2.1 Effettuare il login

- **Attori primari:** Amministratore non autenticato;
- **Precondizioni:** Aprire la pagina di gestione di Umbraco;
- **Postcondizioni:** L'amministratore viene autenticato ed entra nel *Backoffice*;
- **Scenario principale**
  1. L'amministratore non autenticato apre il pannello di Umbraco;
  2. L'amministratore non autenticato inserisce le credenziali;
  3. L'amministratore viene autenticato ed entra nel *Backoffice*.
- **Scenario secondario**
  1. L'amministratore non autenticato inserisce delle credenziali errate.

UC2.2 Aggiornare i dati principali della struttura

* **Attori primari:** Amministratore;
* **Precondizioni:** Aprire la tab contenuti nel *Backoffice* Umbraco;
* **Postcondizioni:** L'amministratore aggiorna i dati principali della struttura;
* **Scenario principale**
  1. L'amministratore apre il menu "settings";
  2. Aggiorna i dati principali della struttura;
  3. Salva e pubblica la pagina con i dati aggiornati.
* **Scenario secondario**
  1. L'amministratore compila dei campi in modo errato (se sottoposti a validazione) o non ne compila di obbligatori;
  2. Viene visualizzato un messaggio di errore.

UC2.3 Pubblicare una nuova sezione

* **Attori primari:** Amministratore;
* **Precondizioni:** Aprire la tab contenuti nel *Backoffice* Umbraco;
* **Postcondizioni:** L'amministratore crea una nuova sezione fra quelle predisposte;
* **Scenario principale**
  1. L'amministratore preme sui tre puntini "..." sulla icona della HomePage;
  2. Sceglie che tipo di contenuto creare;
  3. Compila i campi richiesti;
  4. Salva e pubblica la nuova pagina.
* **Scenario secondario**
  1. L'amministratore compila dei campi in modo errato (se sottoposti a validazione) o non ne compila di obbligatori;
  2. Viene visualizzato un messaggio di errore.

UC2.4 Aggiornare i dati di una pagina creata

* **Attori primari:** Amministratore;
* **Precondizioni:** Aprire la tab contenuti nel *Backoffice* Umbraco;
* **Postcondizioni:** L'amministratore aggiorna i dati della pagina;
* **Scenario principale**
  1. L'amministratore trova nel menu la pagina da aggiornare;
  2. Aggiorna i dati dei campi;
  3. Salva e pubblica la pagina con i dati aggiornati.

* **Scenario secondario**

    1. L'amministratore compila dei campi in modo errato (se sottoposti a validazione) o non ne compila di obbligatori;
    2. Viene visualizzato un messaggio di errore.

UC2.5 Pubblicare una nuova notizia

* **Attori primari:** Amministratore;
* **Precondizioni:** Aprire la tab contenuti nel *Backoffice* Umbraco e aver creato una sezione di novità;
* **Postcondizioni:** L'amministratore crea una nuova notizia da pubblicare;
* **Scenario principale**

    1. L'amministratore entra nel menu delle novità;
    2. Preme sul pulsante "Crea notizia";
    3. Compila i campi richiesti;
    4. Salva e pubblica la nuova pagina.

* **Scenario secondario**

    1. L'amministratore compila dei campi in modo errato (se sottoposti a validazione) o non ne compila di obbligatori;
    2. Viene visualizzato un messaggio di errore.

UC2.6 Pubblicare un nuovo luogo

* **Attori primari:** Amministratore;
* **Precondizioni:** Aprire la tab contenuti nel *Backoffice* Umbraco e aver creato una sezione di luoghi di interesse;
* **Postcondizioni:** L'amministratore crea la pagina di un luogo;
* **Scenario principale**

    1. L'amministratore entra nel menu della sezione dei luoghi di interesse;
    2. Preme su "Crea luogo";
    3. Compila i campi richiesti;
    4. Salva e pubblica la nuova pagina.

* **Scenario secondario**

    1. L'amministratore compila dei campi in modo errato (se sottoposti a validazione) o non ne compila di obbligatori;
    2. Viene visualizzato un messaggio di errore.

UC2.7 Pubblicare una nuova pagina di Orari o contatti **

*Questo UC, come altri segnati da un "**", offre un caso nel dettaglio dell'use case sintetico UC2.16.*

* **Attori primari:** Amministratore;
* **Precondizioni:** Aprire la tab contenuti nel *Backoffice* Umbraco e aver creato una sezione di orari e contatti;
* **Postcondizioni:** L'amministratore crea una nuova pagina di orari e contatti;
* **Scenario principale**
  1. L'amministratore entra nel menu della sezione di orari e contatti;
  2. Preme sul pulsante "Crea Standard Page";
  3. Compila i campi richiesti;
  4. Salva e pubblica la nuova pagina.
* **Scenario secondario**
  1. L'amministratore compila dei campi in modo errato (se sottoposti a validazione) o non ne compila di obbligatori;
  2. Viene visualizzato un messaggio di errore.

UC2.8 Pubblicare una nuova pagina di Tariffa e modulistica **

* **Attori primari:** Amministratore;
* **Precondizioni:** Aprire la tab contenuti nel *Backoffice* Umbraco e aver creato una sezione di tariffa e modulistica;
* **Postcondizioni:** L'amministratore crea una nuova pagina di tariffa e modulistica;
* **Scenario principale**
  1. L'amministratore entra nel menu della sezione di Tariffa e modulistica;
  2. Preme sul pulsante "Crea Standard Page";
  3. Compila i campi richiesti;
  4. Salva e pubblica la nuova pagina.
* **Scenario secondario**
  1. L'amministratore compila dei campi in modo errato (se sottoposti a validazione) o non ne compila di obbligatori;
  2. Viene visualizzato un messaggio di errore.

UC2.9 Pubblicare una nuova pagina di "Chi siamo" **

* **Attori primari:** Amministratore;
* **Precondizioni:** Aprire la tab contenuti nel *Backoffice* Umbraco e aver creato una sezione di "Chi siamo";
* **Postcondizioni:** L'amministratore crea una nuova pagina di "Chi siamo";
* **Scenario principale**
  1. L'amministratore entra nel menu della sezione di "Chi siamo";
  2. Preme sul pulsante "Crea Standard Page";
  3. Compila i campi richiesti;
  4. Salva e pubblica la nuova pagina.
* **Scenario secondario**
  1. L'amministratore compila dei campi in modo errato (se sottoposti a validazione) o non ne compila di obbligatori;
  2. Viene visualizzato un messaggio di errore.

UC2.10 Pubblicare una nuova pagina standard senza padre

* **Attori primari:** Amministratore;
* **Precondizioni:** Aprire la tab contenuti nel *Backoffice* Umbraco;
* **Postcondizioni:** L'amministratore crea una nuova pagina standard;
* **Scenario principale**
  1. L'amministratore preme sui tre puntini "..." di fianco all'icona della Home-Page;
  2. Preme su "Standard Page";
  3. Compila i campi richiesti;
  4. Salva e pubblica la nuova pagina.
* **Scenario secondario**
  1. L'amministratore compila dei campi in modo errato (se sottoposti a validazione) o non ne compila di obbligatori;
  2. Viene visualizzato un messaggio di errore.

UC2.11 Pubblicare una nuova pagina di Persona

* **Attori primari:** Amministratore;
* **Precondizioni:** Aprire la tab contenuti nel *Backoffice* Umbraco e aver creato una sezione di personale amministrativo;
* **Postcondizioni:** L'amministratore crea una nuova pagina di persona;
* **Scenario principale**

1. L'amministratore entra nel menu della sezione del personale amministrativo;
2. Preme sul pulsante "Crea Persona";
3. Compila i campi richiesti;
4. Salva e pubblica la nuova pagina.

* **Scenario secondario**

1. L'amministratore compila dei campi in modo errato (se sottoposti a validazione) o non ne compila di obbligatori;
2. Viene visualizzato un messaggio di errore.

UC2.12 Pubblicare una nuovo argomento/tipologia/categoria

Vengono sintetizzati in un unico UC data la ridondanza

* **Attori primari:** Amministratore;
* **Precondizioni:** Aprire la tab contenuti nel *Backoffice* Umbraco e aver creato una sezione di argomento/tipologia/categoria;
* **Postcondizioni:** L'amministratore crea una nuova pagina di argomento/tipologia/categoria;
* **Scenario principale**

1. L'amministratore entra nel menu della sezione di argomenti/tipologie/categorie;
2. Preme sul pulsante "Crea argomento/tipologia/categoria";
3. Compila i campi richiesti;
4. Salva e pubblica la nuova pagina.

* **Scenario secondario**

1. L'amministratore compila dei campi in modo errato (se sottoposti a validazione) o non ne compila di obbligatori;
2. Viene visualizzato un messaggio di errore.

UC2.13 Pubblicare una nuova pagina con collegamento esterno

* **Attori primari:** Amministratore;
* **Precondizioni:** Aprire la tab contenuti nel *Backoffice* Umbraco;
* **Postcondizioni:** L'amministratore crea una nuova pagina con collegamento esterno;
* **Scenario principale**

1. L'amministratore preme sui tre puntini "..." di fianco all'icona della Home-Page;
2. Preme su "Pagina esterna";
3. Compila i campi richiesti;

4. Salva e pubblica la nuova pagina.

* **Scenario secondario**

1. L'amministratore compila dei campi in modo errato (se sottoposti a validazione) o non ne compila di obbligatori;
2. Viene visualizzato un messaggio di errore.

Ulteriori use case non legati al sito della struttura anziani

UC2.14 Pubblicare la pagina di un ufficio

* **Attori primari:** Amministratore;
* **Precondizioni:** Aprire la tab contenuti nel *Backoffice* Umbraco e aver creato una sezione di amministrazione;
* **Postcondizioni:** L'amministratore crea la pagina di un ufficio;
* **Scenario principale**

1. L'amministratore entra nel menu della sezione di amministrazione;
2. Preme sul pulsante "Crea ufficio";
3. Compila i campi richiesti;
4. Salva e pubblica la nuova pagina.

* **Scenario secondario**

1. L'amministratore compila dei campi in modo errato (se sottoposti a validazione) o non ne compila di obbligatori;
2. Viene visualizzato un messaggio di errore.

UC2.15 Pubblicare la pagina di un servizio

* **Attori primari:** Amministratore;
* **Precondizioni:** Aprire la tab contenuti nel *Backoffice* Umbraco e aver creato una sezione di servizi;
* **Postcondizioni:** L'amministratore crea una nuova pagina di servizio;
* **Scenario principale**

1. L'amministratore entra nel menu della sezione dei servizi;
2. Preme sul pulsante "Crea servizio";
3. Compila i campi richiesti;
4. Salva e pubblica la nuova pagina.

* **Scenario secondario**

1. L'amministratore compila dei campi in modo errato (se sottoposti a validazione) o non ne compila di obbligatori;
2. Viene visualizzato un messaggio di errore.

UC2.16 Pubblicare una nuova pagina di informazione standard

*Questo UC potrebbe sintetizzare tutti quelli relativi alle standard page, riportati nel dettaglio per chiarezza informativa e segnati da un "**".*

* **Attori primari:** Amministratore;
* **Precondizioni:** Aprire la tab contenuti nel *Backoffice* Umbraco e aver creato una sezione standard;
* **Postcondizioni:** L'amministratore crea una nuova notizia da pubblicare;
* **Scenario principale**
    1. L'amministratore entra nel menu della sezione standard;
    2. Preme sul pulsante "Crea pagina standard";
    3. Compila i campi richiesti;
    4. Salva e pubblica la nuova pagina.
* **Scenario secondario**
    1. L'amministratore compila dei campi in modo errato (se sottoposti a validazione) o non ne compila di obbligatori;
    2. Viene visualizzato un messaggio di errore.

3.6 Tracciamento dei requisiti

Completati casi d'uso, si è rivelato necessario tracciare i requisiti. Ad ogni requisito individuato è stato assegnato un codice univoco in cui:

* La prima lettera è **R** (requisito);
* La seconda lettera indica il tipo di requisito:
    - **F** per i requisiti funzionali;
    - **Q** per i requisiti qualitativi;
    - **V** per i requisiti di vincolo;
* un numero individua a quale classe dei casi d'uso fa riferimento;
* un numero progressivo indica il requisito all'interno della sezione.

I requisiti segnati da un "*" non sono strettamente collegati alla struttura anziani, ma vengono comunque riportati perché parte dell'intero *Template*.

3.6.1 Requisiti funzionali

Codice	Descrizione	Rilevanza	Fonti
RF1.1	Si deve poter navigare nelle sezioni principali predisposte dall'amministratore del sito	Obbligatorio	UC1.1
RF1.2	Si deve poter visualizzare una sezione novità	Obbligatorio	UC1.1
RF1.3	Si deve poter visualizzare una notizia proposta in novità	Obbligatorio	UC1.1.1
RF1.4	Si devono poter visualizzare pagine di descrizione della struttura	Obbligatorio	UC1.1.2
RF1.5	Si devono poter visualizzare pagine con le tariffe e i moduli della struttura	Obbligatorio	UC1.1.3
RF1.6	Si devono poter visualizzare orari e contatti della struttura	Obbligatorio	UC1.1.4
RF1.7	Si deve poter inviare una segnalazione attraverso un form	Obbligatorio	UC1.2, UC1.2.1
RF1.8	Deve essere predisposto un sistema di ricerca globale nel sito	Obbligatorio	UC1.3, UC1.3.1
RF1.9	Deve essere creato un collegamento all'amministrazione trasparente	Obbligatorio	UC1.4
RF1.10	Deve essere creato un collegamento all'albo pretorio	Obbligatorio	UC1.5
RF1.10	Deve essere predisposta una sezione con i luoghi di interesse utili vicini alla struttura	Obbligatorio	UC1.6, UC1.6.1
RF1.11	Si deve poter valutare il sito seguendo quanto normato dalle linee guida, con domande condizionali	Obbligatorio	UC1.7, UC1.7.1
RF1.12	Si deve poter vedere il personale amministrativo della struttura	Obbligatorio	UC1.8
RF1.13	Si devono vedere tutte le pagine correlate ad uno stesso argomento	Obbligatorio	UC1.9
RF1.14	Si devono vedere tutte le pagine correlate ad una stessa tipologia	Obbligatorio	UC1.10
RF1.15	Si devono vedere tutte le pagine correlate ad uno stessa categoria	Obbligatorio	UC1.11
RF1.16	Si devono poter vedere e scaricare i file allegati in una pagina	Obbligatorio	UC1.12
RF1.17	Si devono poter accedere e vedere, in qualsiasi pagina, gli antenati del contenuto attuale in una breadcrumb dinamica	Obbligatorio	UC1.13
RF1.18*	Si deve poter visualizzare una sezione di amministrazione	Obbligatorio	UC1.1
RF1.19*	Si devono poter vedere gli uffici contenuti nella sezione di amministrazione	Obbligatorio	UC1.1.5
RF1.20*	Si deve poter visualizzare una sezione dedicata ai servizi	Obbligatorio	UC1.1
RF1.21*	Si devono poter vedere i servizi contenuti nella sezione dedicata ai servizi	Obbligatorio	UC1.1.6

Tabella 3.1: Interazione con il sito web

Codice	Descrizione	Rilevanza	Fonti
RF2.1	L'amministratore deve poter autenticarsi nel *Backoffice*	Obbligatorio	UC2.1
RF2.2	L'amministratore deve poter modificare tutti i dati principali della struttura	Obbligatorio	UC2.2
RF2.3	L'amministratore deve poter pubblicare una nuova sezione fra quelle predisposte dal prodotto	Obbligatorio	UC2.3
RF2.4	L'amministratore deve poter aggiornare i dati di una qualsiasi pagina già creata	Obbligatorio	UC2.4
RF2.5	L'amministratore deve poter pubblicare una nuova notizia	Obbligatorio	UC2.5
RF2.6	L'amministratore deve poter pubblicare un nuovo luogo di interesse	Obbligatorio	UC2.6
RF2.7	L'amministratore deve poter pubblicare una nuova pagina di orari o di contatti	Obbligatorio	UC2.7
RF2.8	L'amministratore deve poter pubblicare una nuova pagina di tariffa e modulistica	Obbligatorio	UC2.8
RF2.9	L'amministratore deve poter pubblicare una nuova pagina informativa della struttura, nella sezione "Chi siamo"	Obbligatorio	UC2.9
RF2.10	L'amministratore deve poter pubblicare una qualsiasi pagina standard senza padre	Obbligatorio	UC2.10
RF2.10	L'amministratore deve poter pubblicare una nuova pagina di una persona nel personale amministrativo	Obbligatorio	UC2.11
RF2.11	L'amministratore deve poter creare un nuovo argomento	Obbligatorio	UC2.12
RF2.12	L'amministratore deve poter creare una nuova tipologia	Obbligatorio	UC2.12
RF2.13	L'amministratore deve poter creare una nuova categoria	Obbligatorio	UC2.12
RF2.14	L'amministratore deve poter pubblicare un oggetto che rimanda ad un contenuto esterno	Obbligatorio	UC2.13
RF2.14*	L'amministratore deve poter pubblicare la pagina di un nuovo ufficio	Obbligatorio	UC2.14
RF2.15*	L'amministratore deve poter pubblicare la pagina di un nuovo servizio	Obbligatorio	UC2.15
RF2.16	L'amministratore deve poter pubblicare una qualsiasi pagina standard	Obbligatorio	UC2.16

Tabella 3.2: Gestione del sito web

3.6.2 Requisiti qualitativi

Codice	Descrizione	Rilevanza	Fonti
RQ1.1	Il sito deve essere accessibile ed essere in conformità con le WCAG2.1	Obbligatorio	Scelta interna
RQ1.2	Non devono essere presenti errori di accessibilità	Desiderabile	Scelta interna
RQ1.3	Il prodotto deve essere continuamente testato con strumenti adatti	Obbligatorio	Scelta interna
RQ1.4	Il prodotto deve adeguarsi alle linee guida per i siti della Pubblica Amministrazione	Obbligatorio	Scelta interna
RQ1.5	Le tassonomie devono seguire quelle presenti nell'allegato tecnico per l'architettura dell'informazione per i servizi digitali della PA	Desiderabile	Scelta interna
RQ1.6	Ogni contenuto deve essere dinamico e modificabile dall'utente amministratore finale, amministratore del sito	Obbligatorio	Scelta interna
RQ1.7	Il codice *Frontend*[g] deve essere adeguatamente commentato	Obbligatorio	Scelta interna
RQ1.8	Il codice *Backoffice* deve essere chiaro e in italiano	Obbligatorio	Scelta interna
RQ1.8	Le tecnologie utilizzate devono sempre essere all'ultima versione disponibile	Obbligatorio	Scelta interna
RQ1.8	Il sito deve essere veloce e scalabile su ogni schermo	Obbligatorio	Scelta interna

Tabella 3.3: Requisiti qualitativi

3.6.3 Requisiti di vincolo

Codice	Descrizione	Rilevanza	Fonti
RV1.1	Utilizzo della libreria *Bootstrap Italia* per il *Frontend*	Obbligatorio	Scelta interna
RV1.2	Utilizzo di *Umbraco CMS* per l'intera infrastruttura del prodotto	Obbligatorio	Scelta interna
RV1.3	Il codice deve essere condiviso e reso accessibile ai tecnici sulla repository *GitLab* predisposta ai fini del progetto	Obbligatorio	Scelta interna

Tabella 3.4: Requisiti qualitativi

Capitolo 4

Strumenti e ambiente di sviluppo

In questo capitolo vengono esposte le varie tecnologie e gli strumenti utilizzati ai fini dello sviluppo.

4.1 Umbraco CMS

Umbraco CMS è un Content Management System Open-Source basato sulla più recente tecnologia *.NET* (ASP.NET core[1]).
Permette agli editor una facile e intuitiva gestione dei contenuti, con potenti strumenti di editing di media, file, testi e molto altro.
Predisposto il sistema, permette di pubblicare con facilità nuove pagine web che andranno ad integrarsi automaticamente al sito web di partenza secondo il *Template* e i modelli forniti dagli sviluppatori.
Umbraco permette infatti agli sviluppatori di offrire un'esperienza di editing personalizzata per qualsiasi tipo di contenuto.

Una potente *API* e un *Service Level* lavorano assieme con modelli di contenuto flessibili per garantire un facile apprendimento e una ottima scalabilità.
Vengono utilizzati una *Pipeline di Routing* e il *Models Builder* per eseguire rendering di contenuti in pochissimo tempo; a gestire e semplificare il lavoro ci sono *Controller MVC* e *Web API*.
Sono inoltre disponibili svariati *Plugin* per raffinare e personalizzare il *Backend* in base alle proprie esigenze.

I vantaggi che offre Umbraco rispetto ad altri CMS sono:

* la possibilità di realizzare il sito web da pagine, template completamente vuoti. Questo permette massima libertà allo sviluppatore, che può così creare soluzioni *ad hoc* in base alle esigenze e creando una propria struttura da zero.

* la struttura formata da **tipi di dati**, **template**, e **contenuti inseriti da Backend**, è agilmente implementabile e visualizzabile da codice per mostrare i dati da Frontend. Questo è possibile grazie al linguaggio *Razor*[g] *C#* che

[1] *ASP.NET Core*. URL: https://docs.microsoft.com/it-it/aspnet/core/?view=aspnetcore-6.0.

permette, con facile curva di apprendimento, di creare pagine dinamiche che si interfacciano e prendono forma dal *Backoffice*.

* Ogni pagina su Umbraco può essere associata ad un *template* che cambia la grafica con cui vengono mostrati i dati. Si può inoltre partire da un *Master Template* per creare una forma base da dare ad ogni pagina del sito, inserendo per esempio l'*Header*, il *Footer*, i tag main, la *Head* (dinamica rispetto al tipo di pagina e il suo contenuto) e così via.

* Tutta la struttura di Umbraco è caratterizzata da nodi padre e figlio che ereditano le proprietà, nel classico stile *OOP*. Questo permette di creare pagine padre che fanno riferimento ai nodi figli che via via possono crearsi, aggiornando il proprio contenuto dinamicamente trovando ereditati i dati delle pagine che "contengono".

In questo progetto è stata utilizzata la versione *Release Candidate* di Umbraco v10 per permettere una rapida integrazione delle più recenti tecnologie offerte dalla piattaforma.

4.1.1 Installazione

Per installare Umbraco è necessario disporre dell'ultima versione di *.NET* (ora alla v6.0) e un Database SQL Server.
La procedura è quella che segue:

* Aprire il terminale e inserire il comando `dotnet new -i Umbraco.Templates` ;

* Esegui poi il comando `dotnet new Umbraco -name MyProject` per creare un nuovo progetto;

* Entra nella cartella del progetto contenente il file con estensione `.csproj`

* esegui il comando `dotnet run` per eseguire e fare la build del progetto;

* Sulla console dovrebbe apparire la stringa:
`[10:57:39 INF] Now listening on: https://localhost:44388` ;

* A questo punto aprire il browser all'URL indicato e seguire le procedure di login e riferimento al database Microsoft SQL Server.

4.2 Bootstrap Italia

Bootstrap Italia è una raccolta di strumenti liberi utili allo sviluppo di interfacce grafiche moderne, semplici, veloci e standardizzate.
La libreria, giunta alla versione v2.0.0, è completamente *Open-Source* ed è costruita sulle fondamenta di *Bootstrap 5.1.3*, di cui eredita tutte le funzionalità, componenti, griglie e classi di utilità, personalizzandole secondo le Linee Guida di Design per i siti web della Pubblica Amministrazione.
Il file sorgente scaricabile contiene *CSS*, *Javascript*, fonts, icone e ulteriori librerie di stile facilmente implementabili nel proprio sito.

4.2.1 CSS

Una volta scaricato e decompresso il file, all'interno della cartella `css` sarà presente un file CSS minificato (`bootstrap-italia.min.css`) a cui fare riferimento nella head con `<link rel="stylesheet" href="./bootstrap-italia.min.css" />`.

4.2.2 Javascript

All'interno della cartella `js` saranno invece presenti il file di bundle e i componenti suddivisi in moduli.
Dopo aver copiato i file all'interno del progetto, sarà sufficiente inserire una versione dei tag `<script>` di seguito riportati alla fine della pagina *HTML*, prima della chiusura del tag `</body>`: `<script src="./bootstrap-italia.bundle.min.js"></script>`

4.2.3 Fonts

Si possono inoltre includere i fonts all'interno di un tag `<script>` con:
`window.__PUBLIC_PATH__ = /bootstrap-italia/dist/fonts`

4.2.4 Librerie di terze parti

Bootstrap Italia include inoltre varie librerie di terzi per facilitare lo sviluppo grafico. Queste sono:

* Splide[2]: per la creazione di caroselli *responsive* e accessibili;
* Masonry[3]: per la creazione di gallerie di immagini responsive e animate;
* ImageSharp[4]: per la gestione veloce e inline del processing delle immagini.

4.3 Visual Studio Code e Visual Studio

Per la corretta gestione di Umbraco, si richiede di utilizzare **Visual Studio** o in alternativa **Visual Studio Code**, perdendo alcune funzionalità.

4.3.1 Visual Studio Code

Visual Studio Code[5] è un editor di codice sorgente sviluppato da Microsoft per Windows, Linux e macOS. Include il supporto per debugging, un controllo per Git integrato, syntax highlighting, IntelliSense, snippet e refactoring del codice. Nel corso del progetto il codice è stato scritto principalmente con questo editor, mantenendolo aggiornato sempre all'ultima versione disponibile.

[2] *SplideJS*. URL: https://splidejs.com/.
[3] *Masonry*. URL: https://masonry.desandro.com/.
[4] *ImageSharp*. URL: https://sixlabors.com/products/imagesharp/.
[5] *Visual Studio Code*. URL: https://code.visualstudio.com/.

4.3.2 Visual Studio

Visual Studio[6] è un ambiente di sviluppo integrato sviluppato da Microsoft.
Visual Studio è multilinguaggio e attualmente supporta la creazione di progetti per varie piattaforme, tra cui anche mobile e console. È possibile creare ed utilizzare estensioni e componenti aggiuntivi.
In questo progetto è stato utilizzato per permettere una più rapida gestione dei pacchetti `NuGet` con cui sono distribuiti Umbraco CMS e le utility che fornisce; anche in versioni beta.
È stato necessario inoltre per creare i file di configurazione per la pubblicazione del sito web sui domini; operazione non possibile con Visual Studio Code.

4.4 Database

Umbraco CMS richiede un database **Microsoft SQL Server**[7].
Questo è stato fornito dall'azienda ospitante, garantendo l'accesso ai server e alle risorse per salvaguardare il contenuto del sito.

4.5 GitLab

Per il Version Control System è stato utilizzato **GitLab**[8].
GitLab è una piattaforma web open source pubblicata nel 2011 che permette la gestione di repository Git e di funzioni trouble ticket; di proprietà della società GitLab Inc.
È stata creata una repository privata, garantendomi l'accesso come *maintainer*, per versionare il codice prodotto giornalmente.
Non sono state utilizzate particolari modalità di workflow, bensì mi è stata lasciata libertà di gestire il progetto in maniera autonoma.

[6] *Visual Studio*. URL: https://visualstudio.microsoft.com/it/.
[7] *Microsoft SQL Server*. URL: https://www.microsoft.com/it-it/sql-server.
[8] *GitLab*. URL: https://gitlab.com.

Capitolo 5

Processo di sviluppo

In questo capitolo viene descritto il processo di sviluppo, analizzando le parti che compongono il prodotto finale.

5.1 Premessa

Il processo di sviluppo viene di seguito descritto suddividendolo nelle varie parti che compongono il Template. Partendo da una analisi delle componenti fondamentali che compongono il Back-end, si passerà a vedere come queste hanno costruito le varie sezioni principali del prodotto, analizzandone le proprietà, la struttura e il comportamento. Ogni parte del Template è stata creata seguendo in modo puntuale e preciso le Linee Guida per i servizi digitali della Pubblica Amministrazione, facendo forte riferimento all'allegato tecnico dell'Architettura dell'informazione[1] sul modello dei Comuni Italiani per trarre i *Content Type* comuni ad ogni sito della PA e le tassonomie di riferimento per adeguare il *Backoffice* alle norme.

Il seguente capitolo illustrerà lo sviluppo del Template per la Pubblica Amministrazione applicandolo ad un caso particolare, di supporto alla costruzione del progetto. Le immagini dell'aspetto grafico, pertanto, saranno rese tramite l'applicazione di quanto sviluppato al sito della casa per anziani di Auronzo Di Cadore.

5.2 Document Types, templates, elements e Compositions

5.2.1 Document Types e Data Types

Fra i primi concetti fondamentali di Umbraco troviamo quello di **Document Type**: la creazione di questo è il primo passo per iniziare lo sviluppo di un sito o di una nuova pagina di contenuto.
Un *Document Type* è un contenitore di dati in cui si possono aggiungere **proprietà**, cioè campi dati e/o attributi associati ad un valore.

[1] *Architettura dell'informazione Designers Italia.* URL: https://docs.google.com/spreadsheets/d/1D4KbaA__xO9x_iBm08KvZASjrrFLYLKX/edit#gid=2066775910.

Ogni proprietà ha un *Data Type*[g], ovvero un tipo di dato, selezionabile fra i molteplici forniti dal CMS (come stringhe di testo, numeriche, RTF, dropdown menu, nested element e molti altri).
In aggiunta si possono creare ulteriori *Data Type* partendo da quelli base, nonché aggiungerne altri da plugin esterni.
Generato un *Document Type*, i suoi dati vengono messi in output tramite un **template**.

Per generare un nuovo *Document Type* (Figura 5.1) si deve andare nel Menu *Settings*, selezionare i tre punti "..." di fianco alla sezione apposita e scegliere "*Document Type with template*".

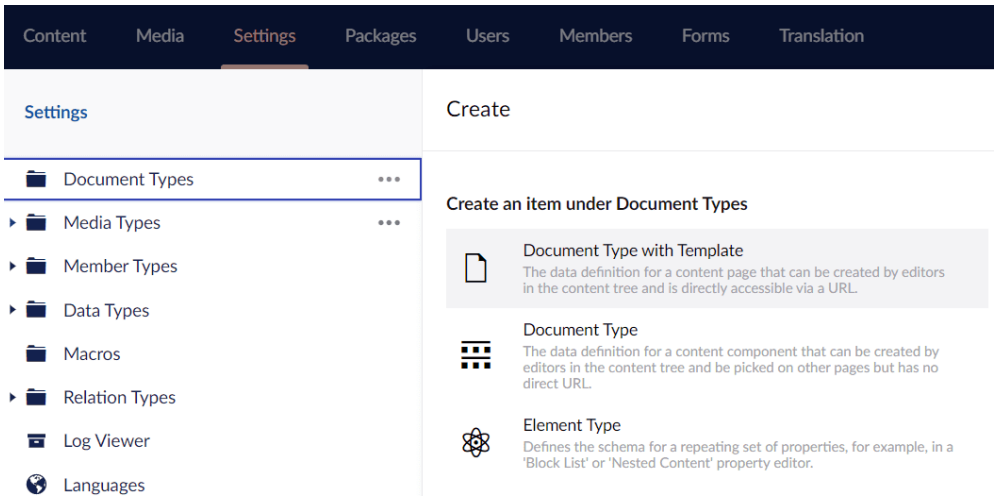
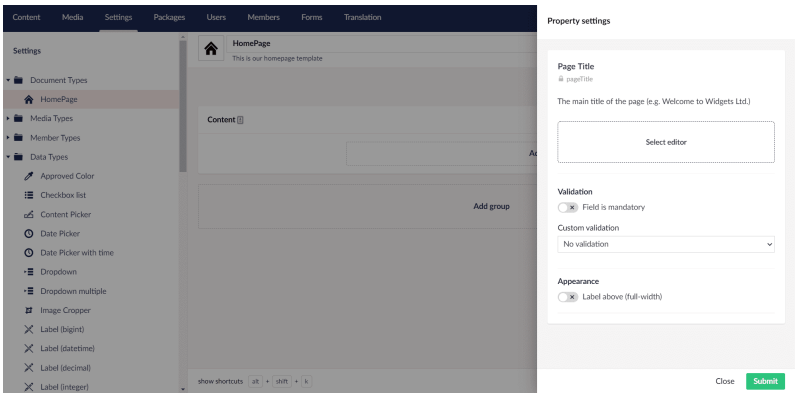
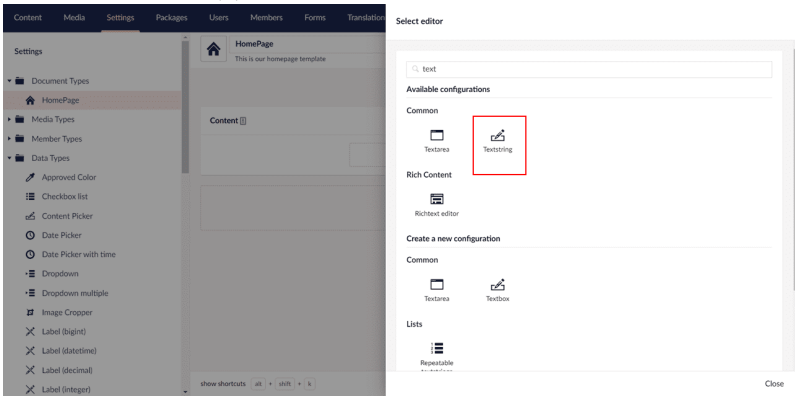


Figura 5.1: Creazione di un nuovo *Document Type*

Una volta inserito il nome, verrà generato in automatico un **alias**, eventualmente modificabile. Quest'ultimo è fondamentale per far riferimento al *Document Type* nel codice sorgente, così come le proprietà che lo compongono.
Dopo aver inserito dei parametri opzionali (come descrizione e Tab per dividere il contenuto in maniera più organizzata) si possono creare dei **gruppi** in cui inserire le proprietà (Figura 5.2a).
Cliccando quindi su "*Add property*" si inizia il processo di creazione di un nuovo dato. Da qui, premendo su "*Select Editor*" si può scegliere un *Data Type* (Figura 5.2b) fra quelli già predisposti o in alternativa modificarne uno a piacimento variandone i parametri (come lunghezza massima, valori di default, limiti e contenuto). Si può scegliere inoltre se rendere quella proprietà obbligatoria o meno, con un messaggio di errore personalizzabile.



(a) Creazione di una nuova proprietà



(b) Selezione di un *Data Type*

Figura 5.2: Proprietà del *Document Type*

Allo stesso modo di prima, verrà creato un *alias* riferibile all'interno del codice sorgente e, in base al *Data Type* selezionato, questo verrà manipolato in diversi modi tramite linguaggio *Razor*. Quest'ultimo inoltre permette la tipizzazione a runtime, così da utilizzare metodi specifici sulla natura del dato. Ogni *Data Type* e il suo utilizzo viene descritto nella documentazione[2] di Umbraco, con esempi pratici e annesse librerie da includere.

Salvato quindi il *Document Type*, ad esso verrà associato in automatico un **template**, al fine di creare, così, un **nodo** di contenuto.

5.2.2 Template

Come appena accennato, alla creazione di *Document Type* corrisponde la generazione di un file associato, chiamato template.
Il template in sè, è la pagina in formato `.cshtml` (CSharpHTML) che viene utilizzata per renderizzare il contenuto del documento.
Questa pagina sarà quindi gestita in maniera analoga ad un normale file HTML con l'aggiunta di codice Razor (Figura 5.3b) per generarne le parti dinamiche.
In testa al file ci saranno i contenuti aggiuntivi importati con la clausola `@using` e le

[2] *Our Umbraco Documentation*. URL: https://our.umbraco.com/documentation/.

inclusioni con `@include`.
Si noti bene che ogni parte di codice in Razor è preceduta dal simbolo `"@"`; questo serve a distinguerlo dal normale linguaggio di markup.

Riferendosi quindi al *Document Type*, all'interno di ogni template si possono raccogliere i dati delle proprietà e tipizzarli di conseguenza, applicando logiche e operazioni tipiche di ogni linguaggio di programmazione (`for, foreach, if, else, switch` ecc.).
Ogni template fa riferimento al contenuto del proprio nodo con la clausola `@Model` e per riferirsi ad una determinata proprietà, con dato alias, si utilizza tipicamente `@Model.Value("alias")` oppure `@Model.Value<Type>("alias")` per tipizzare il valore a runtime.
Sono inoltre messi a disposizione molti metodi utili, primi fra tutti quelli che legano un nodo a rispettivi padri e figli, rispettivamente `@Model.Children()` e `@Model.Parents()`. Questi permettono di raccogliere dinamicamente anche i dati dalle pagine che sono logicamente collegate a quella attualmente in lavorazione permettendo di mostrare, per esempio, una breadcrumb in modo dinamico e dei link alle pagine figlie che vengono via via create.

A tutto questo ovviamente si possono includere fogli di stile e script personalizzati, permettendo massima libertà nella resa dei contenuti.

5.2.3 Master Template

Nel momento in cui vogliamo che alcune parti del nostro codice siano riutilizzate all'interno del dominio, serve avere un *layout di base* chiamato **Master Template**.
In questo documento vengono dunque inseriti componenti come l'Header o il Footer (quali possono essere dei file *Partial View* da includere a loro volta), il tag `<head>` e il `<body>`.
Una volta impostato, questo va dichiarato come layout di riferimento nei templates che ne vogliono ereditare le proprietà, cambiando (in testa ad ogni file) la variabile `layout` a, per esempio, `Master.cshtml`.
Il comando Razor `@RenderBody()` (Figura 5.3a), all'interno dei tag `<body>`, permetterà infine di unire dinamicamente il codice dei due file collegati.

5.2.4 Composition

In Umbraco si possono definire delle **composition**, cioè gruppi di proprietà che possono poi essere inclusi nella definizione di molteplici *Document Type*. Nel caso ci siano, quindi, una serie di caratteristiche che devono accomunare più pagine, è utile crearsi una composition per poi inserirla in ognuna di esse, generando una Tab all'interno del *Backoffice*.
Tale oggetto può avere anch'esso nome, proprietà con alias, tab multiple e così via.
In questo progetto è stata creata, fra le molteplici, una composition "*SEO*" con all'interno due proprietà: *Title* e *Description*. In questo modo è stato possibile aggiungere ad ogni contenuto una tab SEO modificabile in base alla pagina che andrà a crearsi. I campi dati correlati vengono quindi popolati dinamicamente grazie al codice Razor.

5.2.5 Element

Un **element** definisce lo schema per un insieme ripetuto di proprietà, ad esempio, in un editor "Block list" o "Nested content".
In questo progetto ne sono stati creati vari esempi, come "*Orari*", "*Telefono*", "*Email*", "*Indirizzo*" e altri.
Questi oggetti, quindi, sono stati utilizzati all'interno di strutture dati come il "Nested Content", che permette di crearne una lista.
Si pensi agli orari di un ufficio: in questo modo un singolo elemento "Orario", composto dalla scelta del giorno della settimana e degli orari giornalieri, è ripetibile in una lista creando, alla fine, una singola proprietà con valori multipli.
In questo modo si dà la possibilità, ad ogni contenuto che lo richiede, di avere il proprio elenco personalizzabile di orari settimanali da renderizzare.

```
@using Umbraco.Cms.Web.Common.PublishedModels;
@using Umbraco.Forms.Web;
@inherits Umbraco.Cms.Web.Common.Views.UmbracoViewPage
@{
    Layout = null;
    var title = Model.Value("metaTitle") ?? Model.Name;
    var description = Model.Value("metaDescription") ?? string.Empty;
}

<!DOCTYPE html>
<html lang="it">
<head>
    <meta name="robots" content="noindex" />
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="description" content="@description">
    <meta name="author" content="Comune di BVB">
    <meta name="title" content="@title">
    <title>@title</title>
    <link rel="stylesheet" href="/assets/bootstrap-italia-v2/css/bootstrap-italia.min.css">
</head>

<body>
    <partial name="Header" />
    <main id="contenutoPrincipale" >
        @RenderBody()
    </main>
    <partial name="Footer" />

    <!-- Fonts e inclusione bundle bootstrap italia -->
    <script>window.__PUBLIC_PATH__ = '/assets/bootstrap-italia-v2/fonts'</script>
    <script src="/assets/bootstrap-italia-v2/js/bootstrap-italia.bundle.min.js"></script>
    @Html.RenderUmbracoFormDependencies()
    <script src="/assets/scripts.js"></script>
</body>

</html>
```

(a) Master Template con codice Razor

```
@foreach (var item in Model.Children().OrderByDescending(t => t.Value<DateTime>("dataPubblicazione")).Take(6))
{
    <li class="splide__slide">
        <div class="it-single-slide-wrapper" style="height: 100%;">
            <!-- Start Card -->
            <div class="card-wrapper" style="height: 100%;">
                <div class="card card-bg">
                    <div class="img-responsive-wrapper">
                        <div class="img-responsive">
                            <figure class="img-wrapper">
                            @{ var immagine = item.Value<MediaWithCrops>("immaginePrincipale"); }
                            <img srcset="@immagine?.MediaUrl()?format=webp" src="@immagine?.MediaUrl()" style="
                            aria-hidden="true" alt="@immagine.Value("alt")">
                            </figure>
                            <div class="card-calendar d-flex flex-column justify-content-center">
                                <span class="card-date">@(item.Value<DateTime>("dataPubblicazione").ToString("d
                                <span class="card-day">@(item.Value<DateTime>("dataPubblicazione").ToString("MM
                            </div>
                        </div>
                    </div>
                    <div class="card-body">
                        <div class="d-flex flex-wrap my-2">
                            @foreach (var subitem in item.Value<IEnumerable<IPublishedContent>>("argomenti"))
                            {
                            <a href="@subitem.Url()">
                                <div class="chip chip-simple chip-primary text-truncate">
```

(b) Codice Razor

Figura 5.3: Master template e codice Razor C#

5.3 Settings

Il Template necessita in primo luogo di un modo per inserire i principali dati dell'ente, del Comune, della struttura a cui si applica.
Elementi come il nome, un motto, i dati di contatto, lo stemma/logo, e la scelta dei menù da visualizzare deve essere lasciata libera e compilabile all'utente finale tramite il CMS.
Questo principio di personalizzazione si applica più in generale ad ogni contenuto, ma "**Settings**" è il primo pezzo a cui metter mano per la creazione del sito.
Questo particolare *Document Type* si comporta come una sorta di variabile globale che andrà applicata a tutte le pagine. Esso infatti sarà l'unico altro nodo radice oltre alla HomePage, trattata in seguito.

5.3.1 Back End e proprietà

"Settings" contiene pertanto le seguenti proprietà:

* **Nome**: campo testuale con il nome dell'ente o, come nel caso del progetto, della struttura;
* **Motto**: campo testuale con una breve frase da far apparire al di sotto del nome. Questo andrà automaticamente a nascondersi su uno schermo mobile;
* **Indirizzo del Comune**: campo testuale con l'indirizzo civico della sede;
* **PEC**: campo testuale, dotato di validazione, dove inserire la PEC dell'ente;
* **Email**: campo testuale, dotato di validazione, dove inserire l' email principale dell'ente;
* **Partita IVA e Codice Fiscale**: campi testuali dove inserire P. IVA e Codice Fiscale dell'ente, se presenti;
* **Ente di appartenenza**: selettore URL al link dell'ente di appartenenza (Regione, Provincia autonoma ecc.);
* **Logo**: *Image Media Picker* per inserire lo stemma o il logo;
* **Favicon**: *Media Picker* per selezionare una *favicon*;
* **Social**: selettore multiplo di URL alle pagine dell'ente sulle principali piattaforme di Social Network;
* **Menu**: Nested content che permette di scegliere i nodi da mostrare nella Navbar della Header principale.
  Ogni elemento sarà quindi un *element* "Voce di Menu" nel quale si possono inserire il link al nodo (preso in automatico dal selettore), un'etichetta per sovraccaricare il nome e la possibilità di rendere quella voce di Menu in forma *dropdown*; mostrando quindi a sua volta i nodi figli nella tendina a comparsa. Si può inoltre decidere di mostrare solo alcune voci figlie;

* **Area Personale e Cambio Lingua**: due *Data Type* booleani che impostati a *true* inseriscono nella Slim Header, rispettivamente, un link all'Area Personale e un menu di selezione lingua. Questi campi sono del tutto opzionali e resi tali per eventuali estensioni del prodotto;
* **Collegamenti footer**: selettore multiplo di URL a nodi e link da inserire nella parte terminale del Footer.
  Fra questi devono essere compresi, preferibilmente, i *Credits*, *Privacy Policy*, *Cookie Policy* e la *Dichiarazione di accessibilità*;
* **Cookie Solution**: area di testo utile ad inserire gli script *embedded* per la visualizzazione dei Pop Up sul consenso dei Cookie.

Si precisa che gli indirizzi e i contatti sono gestiti in modo differente rispetto a proprietà simili in altri *Document Type* (dove si utilizzano *element* e *composition*); questo per la loro unicità e per semplificare l'inserimento dei dati nel Frontend.

5.3.2 Dettagli implementativi

"Settings" viene utilizzato principalmente nell' Header e nel Footer, i quali sono entrambi delle *Partial View*, ovvero delle porzioni di codice parziale, appunto, che vengono inserite nel `Master.cshtml`; permettendo così una migliore gestione delle singole parti. In testa ai due file, fra le due parentesi graffe `{...}` dopo le inclusioni, viene dichiarata la variabile "`settings`" come nel Listing 5.1.

```
var settings = Umbraco.ContentAtXPath("//settings").FirstOrDefault();
```

Listing 5.1: Dichiarazione della variabile contente i dati in Settings

Una volta "racchiuso" i dati di "Settings" nella variabile, si possono creare ulteriori variabili che filtrino il contenuto di interesse, in modo da riutilizzarlo poi agilmente all'interno del Markup. Si veda il Listing 5.2.

```
var socials = settings!.Value<IEnumerable<Link>>("socialLinks") ?? new
    List<Link>();
var menu = settings!.Value<IEnumerable<IPublishedElement>>("menuHeader2")
    ?? new List<IPublishedElement>();
```

Listing 5.2: Dichiarazione variabili a partire da settings

In questo caso, avendo cura di *tipizzare* a runtime le variabili, possiamo salvarci i link ai social e i nodi del menu. Quest'ultima variabile `menu` si può per esempio utilizzare per creare dinamicamente le sezioni nella Navbar (Listing 5.3), che andranno ad aggiornarsi e ad aggiungersi dinamicamente in base alle modifiche apportate nel Backoffice.

```
<div class="menu-wrapper">
    <ul class="navbar-nav">
        @foreach (var item in menu)
        {
            var mlink = item.Value<Link>("mainLink");

            <!-- Link Menu standard -->
            <li class="nav-item">
                <a class="nav-link@(mlink.Udi ==
                    @Udi.Create(Constants.UdiEntityType.Document,
                    Model.Key) ? " active" : string.Empty)"
                    href="@mlink.Url" aria-current="page">
                    <span lang="it">@(item.Value("overrideLabel") !=
                        string.Empty ? @item.Value("overrideLabel") :
                        @mlink.Name )</span>
                </a>
            </li>
            }
        }
    </ul>
</div>
```

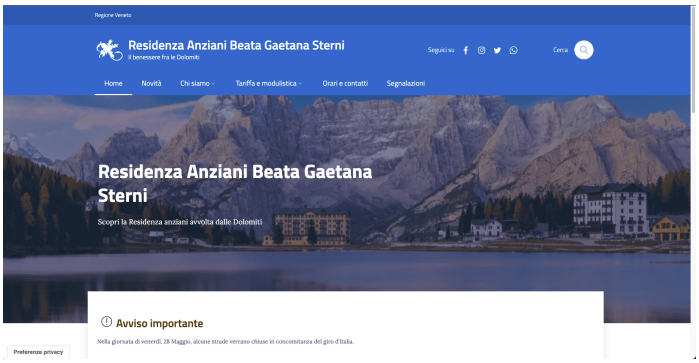
Listing 5.3: Creazione dinamica della Navbar senza dropdown

La proprietà chiamata `Udi` (Listing 5.3) è un identificativo univoco di ogni pagina: questo memorizza tutti i metadati necessari per recuperare un oggetto Umbraco ed è analizzabile all'interno del codice. Come già visto, altre proprietà meno strutturate si possono inserire tramite `settings.Value("alias")`.

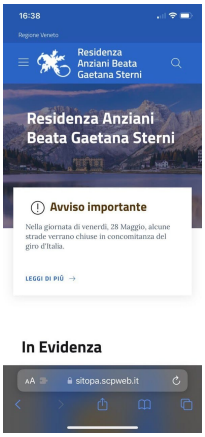
5.4 HomePage

5.4.1 Aspetto

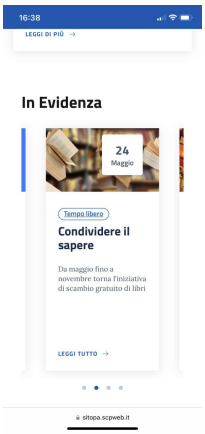
Le immagini fanno riferimento ad un inserimento di contenuti ancora in fase dimostrativa seppur a prodotto finito, pertanto la coerenza di questi potrebbe non essere sempre del tutto ben definita.



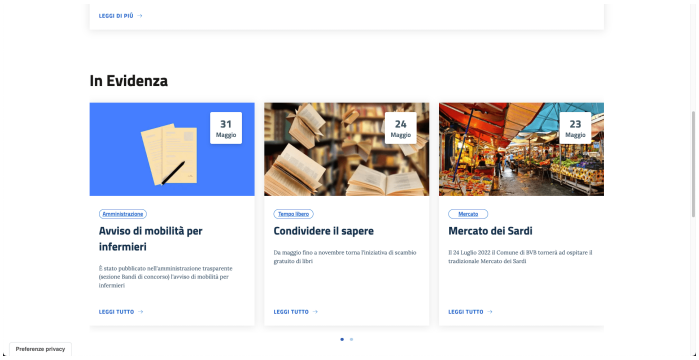
(a) Above the fold da desktop



(b) Above the fold da smartphone



(c) Vista HomePage da smartphone



(d) Vista HomePage da desktop

Figura 5.4: HomePage del sito per la residenza anziani di Auronzo di Cadore

La HomePage è il punto di entrata al sito. Questa è progettata in modo da render subito chiaro all'utente dove si trova, dove può andare e come.
Nella zona *Above The Fold* (Figura 5.4a) vengono presentati l'**Header**, una **Hero** con un messaggio di ingresso, e una preview ad una **Card** che appare in caso di messaggi importanti da segnalare in primo piano.
Da dispositivi mobile si mantiene la stessa presentazione (Figura 5.4b), naturalmente scalando i contenuti.
La Hero, cioè la grande immagine sotto l'Header, viene leggermente oscurata per

permettere una corretta leggibilità del testo sovrastante: questo permette un piacevole effetto di benvenuto alla pagina.
Appena si scorre il contenuto, si trova una sezione dedicata alle novità in evidenza (Figura 5.4c e 5.4d), utile per portare subito in primo piano comunicati importanti facilmente raggiungibili con un solo passaggio. Le *cards* contenute mostrano a colpo d'occhio le principali informazioni, quali la data di pubblicazione, l'argomento di interesse (indicizzabile ad una pagina contenente tutti gli argomenti affini), la descrizione breve della novità e un link rapido ad essa; nonché una immagine di contorno. Il tutto è reso tramite un **Carosello Splide**, per permettere una maggiore orizzontalità del contenuto e una migliore fruizione da mobile. Il carosello, seppur navigabile tramite tastiera e non in movimento automatico, manca delle frecce di navigazione; non implementabili a causa dei limiti imposti dalla libreria.
Sotto la zona in evidenza, infine, si trovano delle card meno strutturate che contengono collegamenti rapidi ad altre pagine o link esterni.

5.4.2 Back End e proprietà

La Homepage contiente, nel Backoffice, le seguenti proprietà:

* **Titolo principale**: campo testuale cui contenuto viene visualizzato in sovrimpressione alla *Hero*. Può essere utile a fornire una vetrina di entrata al sito web;

* **Sottotitolo**: area di testo che contiene, preferibilmente, un breve sottotitolo da mostrare nella *Hero*. Questo andrà automaticamente a nascondersi su uno schermo mobile;

* **Hero Banner**: immagine da visualizzare nella *Hero*. Questa verrà leggermente oscurata per favorire un maggior contrasto con la scritta sovrimpressa. Se l'immagine supera una certa dimensione verrà riadattata per non occupare troppo spazio sullo schermo. Allo stesso modo, se troppo piccola, verrà ingrandita ad un' altezza minima;

* **Testo avviso**: *Rich Text Editor* (abbr. *RTE*[g]) che permette l'inserimento di un avviso da mostrare nella zona *Above The Fold*. Un campo RTE permette una migliore formattazione del contenuto. Se il campo è vuoto, la *Card* non verrà renderizzata;

* **Url Avviso**: selettore di URL che permette di collegare l'avviso importante ad un nodo interno al sito o ad un link esterno. Se, quindi, l'avviso può essere collegato ad un URL che ne possa completare la copertura informativa, è buona prassi inserirlo. Se non compilato, il link rapido non verrà mostrato ai piedi della *Card*;

* **Testo Contenuto principale**: *Grid Layout* da compilare se si desidera mostrare, al di sotto del banner, un contenuto personalizzato aggiuntivo in HomePage. Il **Grid Layout** permette una gestione libera del contenuto, con la possibilità di inserire *RTE*, *Headline*, immagini, file, contenuti e molto altro. Si ottiene in sostanza un editor libero di *Grid* da manipolare come si preferisce. Si decide se avere un layout su una o due colonne, per poi personalizzare l'asse orizzontale impostando una estensione da 1 fino a 12 colonne per ogni riga. Questo campo è

del tutto opzionale e se non inserito verrà mostrato il normale layout per come originariamente concepito;

* **Collegamenti a termine del contenuto**: selettore multiplo di URL che permette di creare delle *Card* a termine della HomePage, prima del Footer, con dei link rapidi a contenuti esterni o nodi interni al sito. Questi sono utili per collegamenti a sezioni come l'*Albo Pretorio* o la *Amministrazione trasparente*, in modo da non avere troppe voci nella NavBar principale;

* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

5.4.3 Dettagli implementativi

La HomePage presenta già molte parti di codice comuni ad altre pagine. In questo caso, si è deciso di inizializzare le variabili già nel momento della creazione della Pagina (Listing 5.4).

```
@{
 Layout = "Master.cshtml";
 var titolo = Model.Value<string>("titoloPrincipale");
 var sottotitolo = Model.Value<string>("sottotitolo");
 var hero = Model.Value<MediaWithCrops>("heroBanner");
 var collegamenti = Model.Value<IEnumerable<Link>>("collegamenti") ?? new
     List<Link>();
 var avviso = Model.Value<string>("testoAvviso");
 var novita = Umbraco.ContentAtXPath("//novita");
}
```

Listing 5.4: Dichiarazione variabili in testa al file

In questo modo tramite degli `if statement` è possibile valutarle e, se non nulle, renderizzare il contenuto HTML all'interno del corpo della condizione (Listing 5.5).

Ad esempio, per mostrare l'avviso importante:

```
@if (!avviso.IsNullOrWhiteSpace())
{
<div class="container">
   <div class="row">
  <div class="col-12">
         <div class="card-wrapper">
           <div class="card card-bg">
              <div class="card-body">
                 <div class="d-flex">
                    <svg class="icon mx-2">
                       <use href="assets/bootstrap-italia/svg/..."></use>
                    </svg>
                    <h2 class="card-title text-warning h4">Avviso
                        importante</h2>
                 </div>
                 <div class="card-text mt-2">
                    @Model.Value("testoAvviso")
                 </div>
                    @{
                       var linkAvviso = Model.Value<Link>("urlAvviso");
                    }
                    @if (linkAvviso != null)
                    {
                    <a class="read-more" href="@linkAvviso.Url">
                       <span class="text">Leggi di pi&uacute;</span>
                       <svg class="icon">
                          <use href="assets/bootstrap-italia/svg/..."></use>
                       </svg>
                    </a>
                    }
    [...]
}
```

Listing 5.5: Rendering solo se viene inserito un avviso da mostrare nalla parte Above The Fold

Si può notare in questo caso come si possano dichiarare delle variabili anche all'interno del codice di Markup; utilizzando la sintassi `@{...}`. In questo modo si possono chiamare delle variabili solo se, in base ad una condizione precedente che le lega logicamente, queste verranno utilizzate o meno.
Si precisa che queste scelte di dichiarazione prefissa o meno non influiscono in modo osservabile sulle prestazioni, grazie alle veloci *API* di rendering interne ad Umbraco.

Un'altra componente molto utilizzata nell'intero dominio è il **Carosello Splide**. Questo permette una disposizione *responsive* e orizzontale dei contenuti, favorendo una maggiore fruibilità dell'interfaccia soprattutto in ottica *mobile first*. Si possono costruire caroselli con immagini, testi e *Card*, come ampiamente fatto nel progetto. Il carosello è dichiarato completamente accessibile da *Bootstrap Italia* e dagli sviluppatori della libreria, ma si vuol far notare la mancanza di un metodo di scorrimento tramite frecce laterali. Non è stato possibile aggiungerle, pertanto il carosello si può scorrere tramite mouse, input touchscreen o tramite dei "pallini" di impaginazione.
Nel caso della HomePage, vengono mostrate le ultime novità nel carosello: qui, il *Document Type* "Novità" non è un figlio diretto.
Anziché il metodo `.Children()`, quindi, viene importato il nodo container tramite `var novita = Umbraco.ContentAtXPath("//novita").`
Questo permette di accedere a contenuti figli e antenati di un nodo senza che vi sia una diretta connessione.

```
@if (novita.Count() > 0)
{
<div class="section">
 <div class="section-content">
  <div class="container my-2">
   <h2>In Evidenza</h2>
   <div class="it-carousel-wrapper it-carousel-landscape-abstract-three-cols
       splide" data-bs-carousel-splide>
    <div class="splide__track">
     <ul class="splide__list">

      @foreach (var item in novita.OrderByDescending(t =>
         t.Value<DateTime>("dataPubblicazione")).Take(6))
       {
        <li class="splide__slide">
         <div class="it-single-slide-wrapper" style="height: 100%;">
          <div class="card-wrapper" style="height: 100%;">
           <div class="card card-bg">
            <div class="img-responsive-wrapper">
             <div class="img-responsive">
              <figure class="img-wrapper">
               @{
                var immagine = item.Value<MediaWithCrops>("immaginePrincipale");
               }
               <img srcset="@immagine?.MediaUrl()?format=webp"
              src="@immagine?.MediaUrl()"
              style="width: 100%; height: 15em; object-fit: cover;"
              aria-hidden="true" alt="@immagine.Value("alt")">
              </figure>
              [...]
```

Listing 5.6: Rendering delle Card novità in HomePage

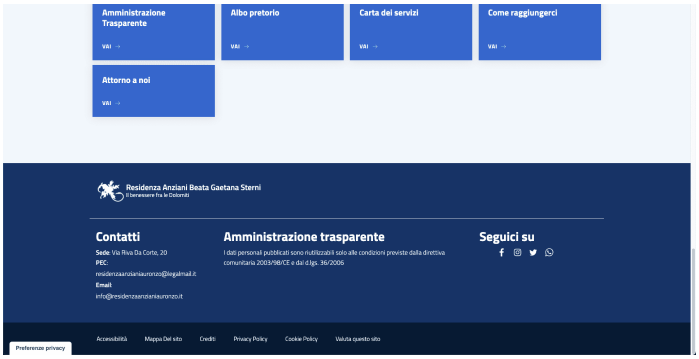
Si possono notare (Listing 5.6) la condizione `if` che innesca il rendering e un `foreach` che permette di accedere agli elementi contenuti nella variabile. In questo caso vengono prelevate le ultime 6 novità ordinandole in modo decrescente rispetto alla data di pubblicazione.
Tramite la variabile `item` che si crea ad ogni iterazione, possiamo accedere ai dati del

singolo nodo, tramite il classico metodo `.Value("alias")`. Nel caso delle immagini si evidenzia la possibilità di accedere al path del media (gestito inoltre con la libreria *ImageSharp*) e all'*alt text*.

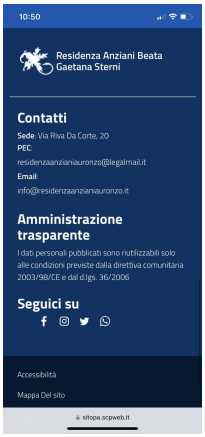
5.5 Header e Footer

5.5.1 Aspetto

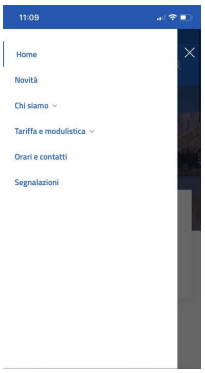
Per le immagini dell'Header fare riferimento alle figure 5.4a e 5.4b nella sezione precedente.



(a) Footer da desktop



(b) Footer da smartphone



(c) Menu laterale da smartphone

Figura 5.5: Header e footer del sito

L'Header e il Footer sono importanti punti di contatto con l'utente.
L'**Header** (Figura 5.4a) contiene il principale menu di navigazione e l'immagine rappresentativa dell'ente. È la prima zona che salta all'occhio dell'utente, e pertanto deve contenere i link alle utilità e le sezioni più importanti. In questo caso l'Header si divide in due zone: una "striscia" più sottile chiamata **Slim Header**, contenente il link all'ente di appartenenza, opzione di cambio lingua e login, e l'**Header principale**, con l'immagine e nome dell'ente, i *social link*, il tasto alla ricerca e la *NavBar principale*. In quest'ultima, si può decidere se rendere le voci dei normali link alle sezioni o dei menu *dropdown*, permettendo di vedere i figli del nodo indicizzato direttamente nella tendina a comparsa. Il modo in cui viene gestita questa scelta nel *Backoffice* è spiegato nei Settings.

Il **Footer** (Figura 5.5a) contiene i principali riferimenti dell'ente, un link veloce all'Amministrazione trasparente e riporta nuovamente l'immagine e i link social del proprietario. Una "striscia" all'estremo pié di pagina, contiene dei link rapidi a nodi o utilità varie. Questa zona è utile per inserire i collegamenti alla *Privacy e Cookie Policy*, dei *Credits*, la dichiarazione di accessibilità e, in questo caso, un link ad una pagina di valutazione del sito.

Laddove da **mobile** (Figura 5.5b) il Footer non subisca particolari modifiche (disponendo semplicemente i contenuti in modo verticale), l'Header si alleggerisce molto dei suoi contenuti (Figura 5.4b), nascondendo o spostando delle sezioni in modo scalato.
Spariscono quindi il sottotitolo e i link social (comunque presenti nel Footer) e le voci di menu vengono raccolte in una schermata laterale richiamabile tramite i tre "trattini", classici delle moderne interfacce mobile. Il tasto per la ricerca viene snellito semplicemente nello stile mantenendo invariata l'icona.

5.5.2 Back End e proprietà

Tutte le proprietà a cui fanno riferimento queste due componenti, sono raccolte nei "Settings", che agiscono come variabile globale. Questo permette di avere invariati e statici i contenuti di Header e Footer in tutte le pagine.

5.5.3 Dettagli implementativi

Queste due componenti sono entrambe delle **Partial View** e vengono importate nel `Master.cshtml`. In questo modo possono essere renderizzate in ogni pagina del sito.
Come descritto in precedenza, prelevano i dati dal *Document Type* "Settings": pertanto, per accedere alle proprietà, bisogna dichiarare la variabile `var settings = Umbraco.ContentAtXPath("//settings").FirstOrDefault()`.
Fatto questo sarà quindi possibile inserire il contenuto in modo "tradizionale".
Un esempio importante è quello della NavBar principale, che offre una discreta difficoltà nella gestione.
Con la variabile `menu` (Listing 5.7) si seleziona quindi la proprietà desiderata.

```
var menu = settings!.Value<IEnumerable<IPublishedElement\>("menuHeader2")
    ?? new List<IPublishedElement>();
```

Listing 5.7: dichiarazione variabile `menu`

Con l'operatore "`??`" si garantisce la presenza di una lista vuota in caso di mancata impostazione.
Si genera il rendering della NavBar come mostrato nel Listing 5.8.

```
<div class="menu-wrapper">
  <ul class="navbar-nav">
    @foreach (var item in menu) {
    var mlink = item.Value<Link>("mainLink");
    <!-- Menu Dropdown -->
    @if(item.Value<bool>("showChildren") && (mlink!.Udi != null)) {
      <li class="nav-item dropdown">
        <a class="nav-link dropdown-toggle" role="button" href="#"
            id="@mlink.Name.Replace(" ","")" data-bs-toggle="dropdown"
            aria-haspopup="true" aria-expanded="false" >
          <span>@(item.Value("overrideLabel") != string.Empty ?
              @item.Value("overrideLabel") : @mlink.Name )</span>
          <svg class="icon icon-xs">
            <use href="/assets/bootstrap-italia/svg/..."></use>
          </svg>
        </a>
        <div class="dropdown-menu" role="region"
            aria-labelledby="@mlink.Name.Replace(" ","")" style="white-space:
            nowrap;">
          <div class="link-list-wrapper">
            <div class="link-list-heading">Scopri</div>
            <ul class="link-list">
            @{
              IEnumerable<IPublishedContent> children = new
                  List<IPublishedContent>();
              if (item.HasValue("linkChildren")) {
              children =
                  item.Value<IEnumerable<IPublishedContent>>("linkChildren");
              }
              else {
                children = Umbraco.Content(mlink.Udi)!.Children;
              }
              }
              @foreach (var subitem in children) {
              <li>
                <a class="list-item" href="@subitem.Url()">
                  <span>@subitem.Name()</span>
                </a>
              </li>
            }
        [...]
    }
    else {
    <!-- Link Menu standard -->
    <li class="nav-item">
      <a class="nav-link@(mlink.Udi ==
          @Udi.Create(Constants.UdiEntityType.Document, Model.Key) ? " active"
          : string.Empty)" href="@mlink.Url" aria-current="page">
        <span lang="it">@(item.Value("overrideLabel") != string.Empty ?
            @item.Value("overrideLabel") : @mlink.Name )</span>
    [...]
```

Listing 5.8: Rendering della NavBar principale *(Il codice è stato appositamente troncato nelle sue parti più essenziali.)*

Si può notare come, in base alla scelta apportata ad ogni voce di menu, il render sia reso condizionale facendo apparire il link come *dropdown* o normale ipertesto (Listing 5.8). Nel caso di *Dropdown*, per ogni `item` si utilizza l'`Udi` per accedere, poi, ai nodi figli. Questi vengono poi renderizzati in lista nella tendina a comparsa, la quale è associata ad una *label* costruita a partire dal nome, eliminando gli spazi.
Molti parametri di tipo *ARIA*[g] (*Accessible Rich Internet Applications*) vengono inseriti nei tag, per favorire l'accessibilità del menu, rendendolo "parlante" e interagibile da tastiera.

Per permettere a chi naviga da tastiera di scorrere velocemente al contenuto principale della pagina o al Footer, sono stati inseriti due *skiplink* prima dello *slim header*, visibili al passaggio tramite tasto *tab* (Figura 5.6 e Listing 5.9)

```
<div class="skiplinks">
    <a class="visually-hidden-focusable" href="#contenutoPrincipale">Vai al
        contenuto principale</a>
    <a class="visually-hidden-focusable" href="#footer">Vai al footer</a>
</div>
```

Listing 5.9: Skiplinks in testa ad ogni pagina

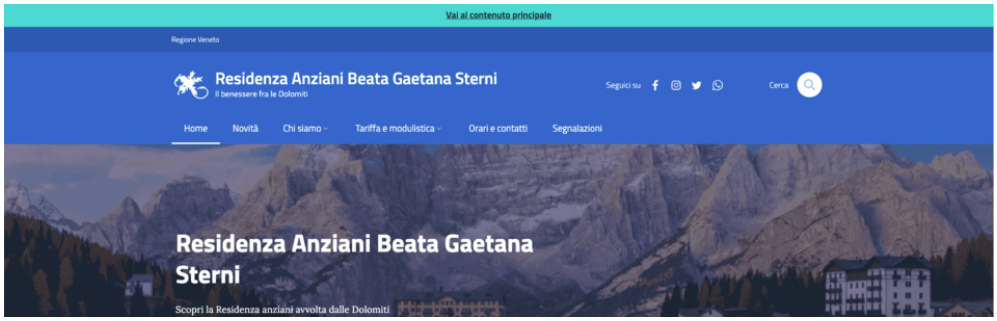
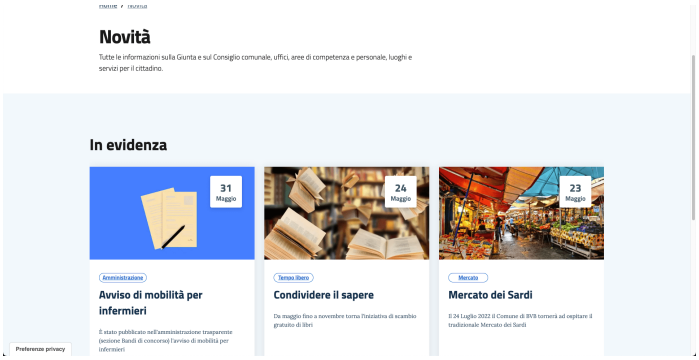


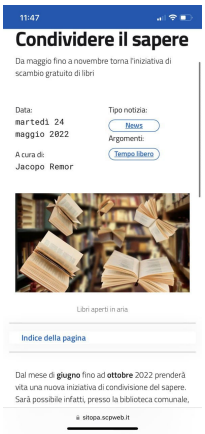
Figura 5.6: Skiplink visibile al passaggio tramite input da tastiera

5.6 Novità

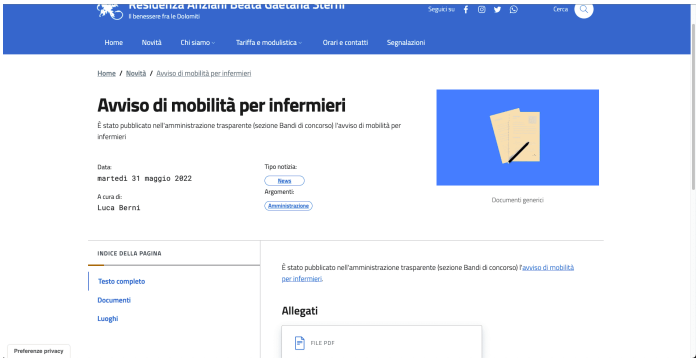
5.6.1 Aspetto



(a) Pagina novità



(b) Notizia da mobile



(c) Notizia da desktop

Figura 5.7: Sezione novità del sito

La struttura di queste pagine permette di raggiungere con pochi click un contenuto desiderato, e di averne una anteprima in una pagina che lo contiene assieme ad altri oggetti affini. Proprio per questo motivo, il modo di disporre i contenuti nella forma "**Container di pagine di tipo** x" e "**Pagina di tipo** x" è piuttosto ricorsivo nel prodotto.
Ogni pagina *Container*, quindi, sarà composta da una intestazione con titolo e sottotitolo che descrivono di cosa si tratta, per poi presentare, al di sotto, i contenuti figli che raccoglie, con una visualizzazione differente in base al tipo.

In questo caso specifico, la pagina Container "**Novità**" (Figura 5.7a) raccoglie le news, gli articoli e i comunicati stampa in evidenza, all'interno di un carosello; esattamente come per la HomePage.
Le notizie meno recenti, invece, verranno mostrate in una sezione sottostante con una struttura *Grid* tipica della libreria *Bootstrap*; sempre con un design a *Card*.

Le **notizie** in sé (Figura 5.7c) presentano una pagina che possa fornire tutti i dettagli circa la redazione del contenuto e dati accessori che possano completare il quadro informativo, come documenti allegati o luoghi di interesse.
Nella intestazione di una notizia (che può essere di tipo News, Evento o Comunicato stampa) si trovano: il titolo corredato da un breve abstract, data, redattore, il tipo di notizia, l'argomento di cui tratta e una immagine. Parte di queste informazioni, inoltre, sono raccolte dalla pagina padre, il *Container*, per renderizzare le *Card* di preview.
L'argomento e il tipo di notizia sono delle *Chip*, cioè piccole "pillole" che se cliccate mostrano tutti i contenuti dello stesso tipo.
All'intestazione segue il corpo della notizia, con eventuali riferimenti utili.
Infine è presente un indice di pagina laterale, che permette di navigare velocemente nel testo della notizia; utile in caso di contenuti molto lunghi.
Il tutto scala su mobile portando ad una maggiore verticalità del contenuto (Figura 5.7b), nascondendo l'indice di pagina in un menu richiamabile al di sotto dell'intestazione (similmente al menu principale dell'header).

5.6.2 Back End e proprietà

Le pagine *Container* presentano una struttura sempre uguale nel *Backend*[g], e quindi nel *Backoffice*. Questo permette di avere una struttura e una gestione dei contenuti sempre uniforme e coesa fra tutte le pagine dello stesso tipo. A cambiare sarà la presentazione nei nodi figli, gestita lato codice.
Le proprietà di "Novità" e "Notizia" sono quindi le seguenti:

Novità

* **Titolo Pagina**: campo di testo con il titolo della pagina di tipo Container, che rispecchia la macro categoria di contenuti che raccoglie;
* **Descrizione Pagina**: area di testo con breve descrizione di cosa si tratta, presentata sotto il titolo;
* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

Notizia

* **Titolo News**: campo di testo con il titolo della notizia;
* **Abstract News**: area di testo con un breve abstract della notizia. Questa proprietà verrà utilizzata anche nella pagina padre per mostrare una anteprima nella Card che la rappresenta;
* **Data pubblicazione**: *Date picker* per inserire la data di pubblicazione della notizia; di default impostata alla data odierna. Questa informazione viene utilizzata dalla pagina padre;
* **Redattore**: campo di testo con il nome del redattore della notizia;
* **Immagine principale**: *Image Media Picker* per selezionare l'immagine principale della notizia, da riportare anche nella pagina padre;

* **Testo completo**: *RTE* per inserire il testo completo della notizia, integrante all'Abstract;
* item **Documenti allegati**: *Media Picker* per selezionare file allegati, visibili in delle card che ne riportano nome, formato e peso;
* **Luogo**: Selettore multiplo di nodi. Permette di scegliere un nodo di tipo "Luogo", che deve essere creato come contenuto, in modo da renderizzarne le principali informazioni in una Card;
* **Argomenti**: *composition* con una *Data List* personalizzata creata tramite il Plugin "*Contentment*". Permette di selezionare uno o più argomenti, che sono di fatto dei nodi che raccolgono tutti i contenuti che vi fanno riferimento. Questo permette di indicizzare quell'argomento, cliccando sulla Chip, e andare in una pagina dedicata. Questa informazione viene utilizzata anche dal nodo padre;
* **Tipologia**: *composition* con una *Data List* personalizzata che permette di scegliere una tipologia di notizia fra News, Eventi e Comunicati stampa. Il comportamento del link creato è analogo all'argomento;
* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

Generata una sezione "Novità", il *Backoffice* è impostato per dare la possibilità di creare dei nodi figli di tipo "Notizia". Questa modalità è identica per tutte le pagine che condividono la medesima struttura (Figura 5.8).

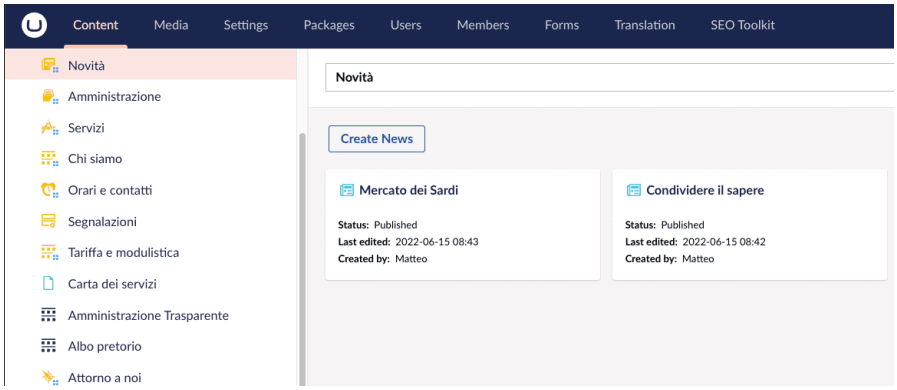


Figura 5.8: Creazione notizie nel backoffice

5.6.3 Dettagli implementativi

Si è gia visto come una pagina possa prelevare dei dati da altri nodi per creare del contenuto dinamico.
In questo caso, i *Document Type* "Notizia" sono impostatati come direttamente nodi figli di "Novità".
Questo permette, oltre che la facile e intuitiva creazione delle pagine nel *Backoffice*, di accedere a tutti i figli tramite il metodo `.Children()`. Il carosello in "Novità", quindi, verrà renderizzato (diversamente dalla HomePage) con una condizione logica come mostrato al Listing 5.10.

```
@foreach (var item in Model.Children().OrderByDescending(t =>
    t.Value<DateTime>("dataPubblicazione")).Take(6))
{
 <li class="splide__slide">
  <div class="it-single-slide-wrapper" style="height: 100%;">
   <!-- Start Card -->
   [...]
```

Listing 5.10: `foreach` sui nodi figli

Le "Notizie" presentano un normale accesso diretto alle proprietà del `Model`, ma alcuni *Data Type* definiscono un comportamento a cui prestare attenzione.
È il caso di "luoghi", "tipo di notizia" e "argomenti", che essendo di fatto dei riferimenti ad un nodo esistente nel sito, sono tipizzati come mostrato al Listing 5.11

```
    var luoghi = Model.Value<IEnumerable<IPublishedContent>>("luogo") ?? new
        List<IPublishedContent>();
    var item in Model.Value<IEnumerable<IPublishedContent>>("tipologie");
    var item in Model.Value<IEnumerable<IPublishedContent>>("argomenti");
```

Listing 5.11: IPublishedContent

Sono quindi dei *Published Content Item*, enumerati.
Come tali, vanno iterati se si vuole accedere alle proprietà contenute (ad esempio con un `foreach`). Una volta dentro lo statement, si utilizzano i dati allo stesso modo di una qualsiasi altra iterazione. Naturalmente questo tipo di oggetti incorporano in sè dei metodi speciali per tornare, ad esempio, il proprio Nome, l'ID o l'URL. Un esempio per il rendering delle Card di un "Luogo" è visibile al Listing 5.12

```
<!-- Sezione luogo/luoghi -->
@if (luoghi.Any())
{
<article id="luoghi" class="it-page-section anchor-offset mt-5">
 <h4>Luogo</h4>
 <div class="row">
  @foreach (var item in luoghi)
  {
   <div class="col-6">
    <div class="card card-teaser border rounded shadow p-4">
     <div class="card-body pr-3">
      <h5 class="card-title">@item.Name</h5>
       <div class="card-text">
        <a href="@item.Url()">
         <span>Ulteriori dettagli</span>
       [...]
```

Listing 5.12: Rendering di un *Card*

In queste pagine si può vedere anche il primo caso di **Breadcrumb**, naturalmente presente in ogni contenuto pubblicato eccetto la HomePage.
Allo stesso modo dei nodi figli, si può accedere ai nodi genitori e agli antenati.
Per renderizzare la breadcrumb, quale è una *partial view* come Header e Footer, si utilizza quindi il codice che segue al Listing 5.13.

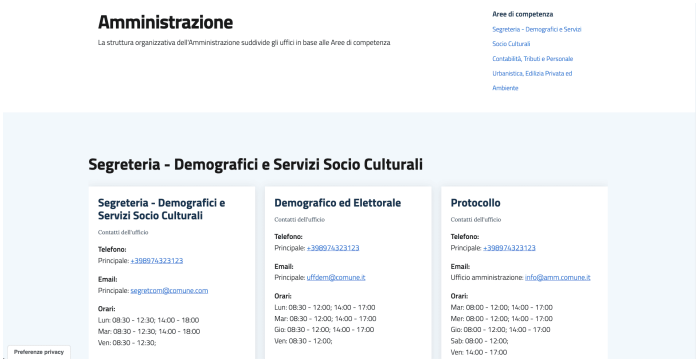
```
@{
    var ancestors = Model.AncestorsOrSelf().Reverse();
}

@if (ancestors != null)
{
<div class="row">
 <div class="col px-lg-4">
  <nav class="breadcrumb-container" aria-label="breadcrumb">
   <ol class="breadcrumb">
    @foreach (var item in ancestors)
    {
     if (item.Id == Model.Id)
     {
      <li class="breadcrumb-item active" aria-current="page">
       <a href="#">@item.Name</a>
      </li>
     }
     else
     {
      <li class="breadcrumb-item">
       <a href="@item.Url()">@item.Name</a><span class="separator">/</span>
      </li>
     }
    }
    [...]
}
```

Listing 5.13: Rendering delle breadcrumb

Si può vedere (Listing 5.13) come si vadano a raccogliere tutti gli antenati del nodo corrente, per poi invertire l'ordine della lista e fare un `foreach` su questa (se non nulla). Facendo un controllo sull'`ID`, inoltre, si può controllare quale sia il link corrente per poter inserire un attributo *aria-current* e creare il giusto rendering secondo la libreria visuale.
Per l'indicizzazione delle *Chip* si rimanda alla sezione dedicata (sezione 5.13).

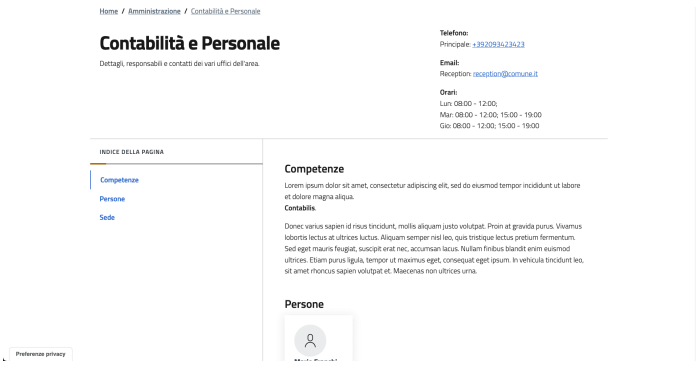
5.7 Amministrazione e uffici

5.7.1 Aspetto



(a) Container amministrazione

(b) Ufficio da mobile



(c) Ufficio da desktop

Figura 5.9: Sezione amministrazione del sito

Queste pagine assumono la stessa struttura delle "Novità".
È quindi presente un container "**Amministrazione**" che raccoglie, suddivisi per categoria di appartenenza, i vari uffici presenti; presentati con un carosello (Figura 5.9a). Qui, la *Card* di ogni ufficio ne riprende nome, contatti inseriti e orari. In questo modo, si possono visualizzare subito i dati salienti del contenuto senza accedervi nel dettaglio, facendo risparmiare tempo all'utente.

La pagina dell'**ufficio** presenta anche qui una struttura già vista (Figura 5.9b), con: intestazione, corpo e indice della pagina. Nell'intestazione è presente il nome, la lista dei contatti e gli orari, resi accessibili per chi fa uso di screen reader.
Il contenuto principale presenta poi un testo che descrive le competenze, corredato eventualmente da "Persone" di rilievo dell'ufficio e la sede di questo; sempre renderizzate tramite una *Card* con collegamento alla pagina apposita.
Come per le notizie e le altre pagine simili, da mobile il contenuto degli uffici viene

fortemente verticalizzato, mentre il carosello nel *container* aiuta a mantenere una resa e una fruizione orizzontale delle sezioni (Figura 5.9c).

5.7.2 Back End e proprietà

Il *Backoffice* della pagina di "Amministrazione" è analogo a quello di "Novità", mentre quello dei singoli uffici presenta delle proprietà specifiche.

Amministrazione

* **Titolo Pagina**: campo di testo con il titolo della pagina di tipo container, che rispecchia la macro categoria di contenuti che raccoglie;
* **Descrizione Pagina**: area di testo con breve descrizione di cosa si tratta, presentata sotto il titolo;
* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

Ufficio

* **Nome Ufficio**: campo di testo con il nome dell'ufficio;
* **Numeri di telefono**: *Nested content* che permette di scegliere i numeri di telefono da mostrare nella sezione dei contatti.
  Ogni elemento sarà quindi un *element* "Telefono" nel quale si possono inserire un nome che ne descriva lo scopo e il numero in sé, validato con una apposita *RegEx*.
  In questo modo si possono mostrare più di un numero di telefono, o nessuno, in base all'ufficio;
* **Emails**: *Nested content* che permette di scegliere le email da mostrare nella sezione dei contatti.
  Ogni elemento sarà quindi un *element* "Email", il cui funzionamento è analogo a quello dei numeri di telefono;
* **Area amministrativa**: menu a tendina che permette di scegliere l'area amministrativa che compete l'ufficio. Questo *Data Type* è modificabile in base alle esigenze aggiungendo o togliendo voci di menu. Questo dato serve alla pagina padre per suddividere le *Card* degli uffici in sezioni diverse in base all'area.
* **Competenze**: *RTE* dove vengono descritte le competenze dell'ufficio;
* **Persone**: selettore multiplo di nodi a contenuti di tipo "Persona", quali devono essere pubblicati e perciò presenti nel sito;
* **Indirizzo**: *composition* che permette di inserire il nome della via, il numero civico, il nome del Comune, sigla della Provincia e CAP della sede. Questa composizione è stata creata per permettere a questa pagina di ereditare proprietà uguali ad altri *Document Type* ;
* **Orari**: *composition* contenente un *Nested content* che permette di scegliere gli orari giornalieri da mostrare nella intestazione. Ogni elemento sarà quindi un *element* "Orari settimanali" che permette di scegliere un giorno della settimana e rispettivi orari di chiusura e apertura. La visualizzazione corretta degli orari è

poi gestita da codice Razor, permettendo anche di rendere accessibili gli orari agli screen reader;

* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

5.7.3 Dettagli implementativi

La rappresentazione del carosello nella pagina "Amministrazione" differisce dalle altre pagine *container* per il modo in cui organizza le *Card*.
Accedendo ai figli, si possono utilizzare dei metodi specifici per raggrupparli in base ad una determinata proprietà. In questo caso si utilizza il metodo `.GroupBy()` per raggruppare gli uffici in base all'area amministrativa. Si può poi accedere al valore della chiave per cui avviene questa procedura tramite il campo `.Key`.
Si dichiara quindi la variabile `areas` come mostrato al Listing 5.14.

```
var areas = Model.Children?.GroupBy(c=>c.Value<string>("areaAmministrativa"))
```

Listing 5.14: Raggruppamento per valore

Successivamente si possono renderizzare le varie sezioni dinamicamente (Listing 5.15).

```
@{ int i = 0; } <!-- variabile di supporto -->
@foreach (var area in areas)
{
i += 1; <!-- incremento della variabile di supporto per alternare le sezioni
    -->
<div class="section@(i%2==1 ? " section-muted" : string.Empty)"
    id="@(LongToShortArea(@area.Key))">
 <div class="section-content">
  <div class="container my-2">
   <h2>@area.Key</h2>
   <div class="it-carousel-wrapper it-carousel-landscape-abstract-three-cols
       splide" data-bs-carousel-splide>
    <div class="splide__track">
     <ul class="splide__list">
      @foreach (var item in area){
   [...]
```

Listing 5.15: Renderizzazione dinamica delle sezioni

In questo caso è stata utilizzata una funzione `LongToShortArea` costruita Ad Hoc per trasformare i nomi delle aree amministrative in una stringa più breve e senza spazi per assegnarlo ad un ID. In questo modo si sono potuti creare degli *skiplink*, nell'intestazione della pagina, per poter navigare più rapidamente alla sezione desiderata.
Una funzione simile chiamata `shortToLongDay` aiuta a dare un `title` al tag `<abbr>` utilizzato negli orari settimanali, per indicare correttamente il giorno della settimana alle tecnologie assistive.
Proprio gli orari subiscono un particolare processo al fine di renderli accessibili e visualizzabili in modo corretto e compatto. Con il codice che segue (Listing 5.16) è stato possibile gestire una visualizzazione sintetica di questi, ma permettendo agli screen reader di leggere e fruire correttamente del contenuto senza perdita di informazione.

```
@{ var dayItems =
    item.Value<IEnumerable<IPublishedElement>>("orariSettimanali"); }
@foreach (var dayItem in dayItems)
{
<p class="my-0 py-0">
 @{ string day =
     shortToLongDay(dayItem.Value<string>("giornoDellaSettimana")); }
  <abbr title="@day">
   @dayItem.Value("giornoDellaSettimana"):
  </abbr>
  @((dayItem.Value<DateTime>("mattinaOrarioInizio").ToShortTimeString()) !=
      "00:00" ?
      dayItem.Value<DateTime>("mattinaOrarioInizio").ToShortTimeString() + "
      - " + dayItem.Value<DateTime>("mattinaOrarioFine").ToShortTimeString()
      + ";" : "")
  @((dayItem.Value<DateTime>("PomeriggioOrarioInizio").ToShortTimeString())
      != "00:00" ?
      dayItem.Value<DateTime>("PomeriggioOrarioInizio").ToShortTimeString() +
      " - " +
      dayItem.Value<DateTime>("PomeriggioOrarioFine").ToShortTimeString() :
      "")
</p>
}
```

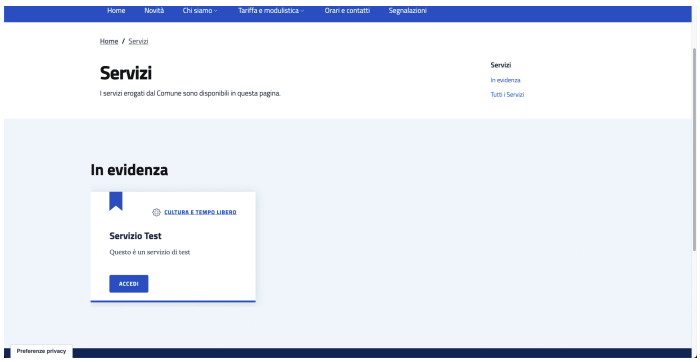
Listing 5.16: Renderizzazione degli orari

Ogni proprietà viene tipizzata a runtime ad un oggetto di tipo `DateTime`, e questo viene manipolato per mostrare le ore in formato "HH:mm". Un orario viene inoltre nascosto se impostato come nullo (in automatico il *Backoffice* assegna un valore alla mezzanotte), e viene inserita o meno della punteggiatura appropriata per suddividere gli orari mattutini da quelli pomeridiani.

5.8 Servizi

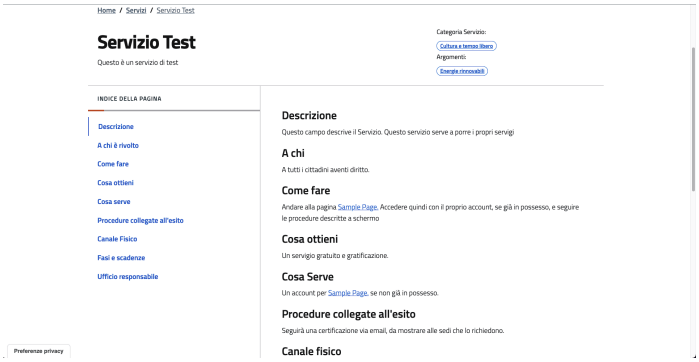
5.8.1 Aspetto

Le pagine che seguono non sono realmente utilizzate nel sito della Residenza anziani, a meno di future estensioni, ma sono state lasciate a scopo dimostrativo.



(a) Container servizi

(b) Servizio da mobile



(c) Servizio da desktop

Figura 5.10: Sezione servizi del sito

La coppia di *Document Type* "Servizi" e "Servizio" seguono la stessa linea operativa delle pagine appena descritte.
Come negli altri casi già discussi, anche qui, in "**Servizi**" cambia la visualizzazione delle *Card* per i contenuti figli, sempre raccolti in un carosello, per quelli in evidenza, e in un sistema di *Grid*, nella sezione sottostante, per tutti gli altri (Figura 5.10a).
In questo caso la singola *Card* è molto più semplice, con un titolo e una descrizione brevissima del servizio; corredati da un link rapido alla categoria di appartenenza. Il bottone di accesso al servizio è reso in modo più marcato, per trasmettere una maggiore attenzione nel momento in cui si naviga all'interno della pagina.
In caso di future estensioni del prodotto, si possono prevedere sistemi di pagamento o di gestione di dati personali all'interno dello stesso dominio; da qui l'importanza di caricare l'accento sul bottone.

La pagina del singolo "**Servizio**" (Figura 5.10c e 5.10b) segue la stessa filosofia delle pagine "Notizia" e "Ufficio" ma qui, in base a quanto descritto nell'allegato per l'architettura dell'informazione, ogni campo è obbligatorio. Per questo motivo, anche il *Backoffice* è impostato in modo molto più rigido, affinché ogni proprietà assuma un valore da mostrare a schermo. In aggiunta ad "Argomento" e "Tipologia", presenti nelle notizie, si trova una terza classificazione del contenuto chiamata "Categoria". Questa è resa specifica per i singoli servizi, ma rimane estendibile a future implementazioni.

5.8.2 Back End e proprietà

Si analizzano di seguito le proprietà dei *Document Type* "Servizi" e "Servizio":

Servizi

* **Titolo Pagina**: campo di testo con il titolo della pagina di tipo Container, che rispecchia la macro categoria di contenuti che raccoglie;
* **Descrizione Pagina**: area di testo con breve descrizione di cosa si tratta, presentata sotto il titolo;
* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

Servizio

* **Nome Servizio**: campo di testo con il nome del servizio;
* **Descrizione breve del Servizio**: campo di testo con una breve descrizione. Questa proprietà viene utilizzata anche dalla pagina padre;
* **Descrizione**: *RTE* con una descrizione più completa del servizio;
* **A chi**: *RTE* dove si descrive a chi si rivolge, inteso come categorie di persone;
* **Come fare**: *RTE* dove si illustrano le procedure da seguire per usufruire del servizio;
* **Cosa ottieni**: *RTE* dove si descrivono uno o più *output* prodotti dal servizio. Ad esempio: "certificato di residenza" o "carta di Tipologia elettronica";
* **Cosa serve**: *RTE* con l'elenco delle cose necessarie per attivare il servizio (documenti necessari, o altro), le relative istruzioni all'uso ed un eventuale link di approfondimento;
* **Procedure esito**: *RTE* con la spiegazione relativa all'esito della procedura, e dove eventualmente ritirarlo (sede dell'ufficio, orari, numero sportello, ecc.);
* **Canale fisico**: *RTE* dove inserire un breve testo che segnala la presenza di una o più sedi dove fruire del servizio. Ad esempio "Il servizio è disponibile in tutte le sedi ... ";
* **Fasi e scadenze**: *RTE* dove indicare data di scadenza del servizio (ad esempio "iscrizione asilo nido entro ..."). Se il servizio è diviso in fasi, prevedere un campo per ciascuna di esse con relativa indicazione dei tempi (ad esempio "iscrizione asilo nido" <data> + "esito della domanda" <data>);

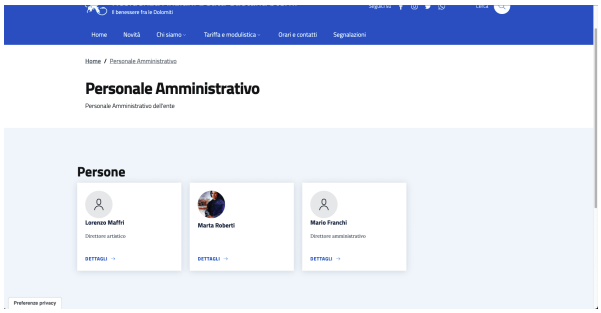
* **Ufficio responsabile**: selettore multiplo di nodi a pagine di tipo "Ufficio", utile a mostrare l'ufficio responsabile dell'erogazione del servizio ;
* **Categoria**: *composition* con una *Data List* personalizzata. Permette di selezionare una categoria di appartenenza. Questo permette, come nel caso di "Argomenti" e "Tipologia" di indicizzare quella categoria, cliccando sulla Chip, e andare in una pagina dedicata. Questa informazione viene utilizzata anche dal nodo padre;
* **Argomenti**: *composition* con una *Data List* personalizzata. Permette di selezionare uno o più argomenti, in modo analogo a quanto accade per le notizie;
* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

5.8.3 Dettagli implementativi

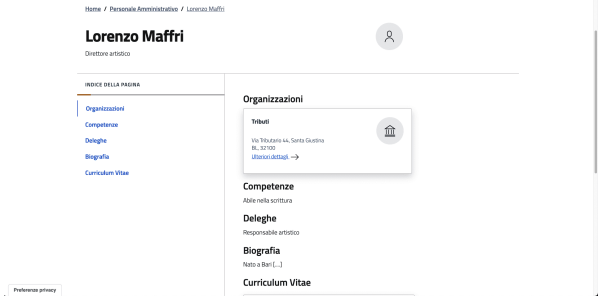
I *template* per i *Document Types* "Servizi" e "Servizio" non presentano particolari caratteristiche, se non quelle già analizzate nei contenuti simili.
Anche qui, nel *Container*, vengono utilizzate le proprietà dei nodi `.Children()` per dare già una *preview* dei singoli servizi, i quali contengono una struttura molto lineare traendo per lo più i valori dal proprio `Model`.
Si prevedono future estensioni di questi *Document Type* , i quali permetterebbero di integrare servizi interni di pagamento, ad esempio PagoPA, e di conseguenza un sistema di *Login* tramite *SPID*. Questo non è stato possibile farlo a causa delle procedure amministrative che ne seguirebbero, ma la struttura è stata ugualmente predisposta.

5.9 Personale amministrativo e Persone

5.9.1 Aspetto



(a) Container personale amministrativo



(b) Pagina di una persona da desktop

Figura 5.11: Sezione personale amministrativo del sito

Del tutto analogo agli altri contenuti analizzati di tipo *Container*-Pagina.
La pagina "**Personale amministrativo**" non presenta un carosello, ma dispone le *Card* in un sistema a griglia (Figura 5.11a). Vengono mostrati il nome, la carica e una foto (di default o a scelta).
In questo caso specifico, tale contenuto è raggiungibile solamente dalla "Mappa del sito" oppure tramite collegamenti ad altri nodi messi maggiormente in evidenza. Naturalmente, se considerato di rilevante importanza, si può decidere di inserire questa sezione in una voce di menu o in un collegamento rapido della HomePage.

"**Persona**" dispone i contenuti nel modo tradizionale già affrontato, ed aggiunge un collegamento all'eventuale ufficio di competenza (Figura 5.11b). Il collegamento fra ufficio e persone deve essere manuale e a cura dell'amministratore del prodotto finale: non è stato infatti possibile trovare una soluzione adeguata per creare questa relazione $[0...n]$ in automatico.

5.9.2 Back End e proprietà

Si analizzano di seguito le proprietà dei *Document Type* "Personale amministrativo" e "Persona":

Personale amministrativo

* **Titolo Pagina**: campo di testo con il titolo della pagina di tipo Container, che rispecchia la macro categoria di contenuti che raccoglie;
* **Descrizione Pagina**: area di testo con breve descrizione di cosa si tratta, presentata sotto il titolo;
* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

Persona

* **Nome**: campo di testo con il nome della persona;
* **Cognome**: campo di testo con il cognome;
* **Foto della persona**: *Image Media Picker* per selezionare una foto della persona, resa poi in forma circolare. Se non inserita verrà inserito di default un *placeholder*;
* **Incarico**: area di testo dove scrivere l'incarico della persona;
* **Competenze**: *RTE* dove si illustrano le competenze. In caso di persona politica, si inserisce una descrizione testuale del ruolo, comprensiva delle deleghe, oppure in caso di persona amministrativa, descrizione dei compiti di cui si occupa;
* **Deleghe**: *RTE* dove si elencano le deleghe a carico della persona;
* **Biografia**: *RTE* dove è possibile inserire una biografia;
* **Curriculum Vitae**: *Media Picker* per selezionare un Curriculum Vitae. Questo *Data Type* è limitato a soli file non multimediali;
* **Uffici**: selettore multiplo di nodi di tipo *Ufficio*. L'utente amministratore del prodotto finale deve aver cura di creare manualmente la giusta relazione $[0...n]$ fra persone e uffici;
* **Numeri di telefono**: *Nested content* che permette di scegliere i numeri di telefono da mostrare nella sezione dei contatti. Procedura del tutto analoga ai nodi di tipo "Ufficio";
* **Emails**: *Nested content* che permette di scegliere le emails da mostrare nella sezione dei contatti;
* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

5.9.3 Dettagli implementativi

Anche qui non si presentano particolari dettagli implementativi degni di nota rispetto ai *template* con uguale struttura. Si noti solamente che gli uffici, non essendo figli diretti del *Document Type* "Persona", vanno inseriti all'interno di una variabile con `var uffici = Model.Value<IEnumerable<IPublishedContent»("uffici")`. In modo simile il file per il Curriculum Vitae, che sarà un oggetto di tipo `IPublishedContent`. Come tutti i file allegati, la resa visiva del documento è resa seguendo quanto indicato nelle linee guida e più in generale nelle regole di accessibilità; mostrando il formato del file e la dimensione, per rendere noto all'utente cosa sta per vedere (Figura 5.12).

```
@if (cv != null)
{
<article id="cv" class="col-lg-8 it-page-sections-container">
 <h3 class="h4">Curriculum Vitae</h3>
  <div class="list-group-item card card-teaser border rounded shadow p-4
      my-2">
   <div class="card-body pr-3">
    <div class="categoryicon-top">
     <svg class="icon icon-sm">
      <use href="/assets/bootstrap-italia/svg/..."></use>
     </svg>
      <span class="text">File @cv.Value("UmbracoExtension")</span>
    </div>
     <a href="@cv.Url()" target="_blank">Curriculum Vitae</a>
     <p>
      [@((Math.Round(Convert.ToDecimal(@Convert.ToInt64(
      cv.Value("UmbracoBytes"))) / 1048576, 1)).ToString() + "MB")]
     </p>
   </div>
  </div>
</article>
}
```

Listing 5.17: Codice per la resa visiva di un allegato

Si può vedere come sia l'estensione che il peso (in Byte) del file vengano presi direttamente dalle proprietà in sé del documento (Listing 5.17), permettendo una visualizzazione dinamica e automatica di queste caratteristiche. In particolare il peso viene manipolato per convertirlo in MegaBytes.

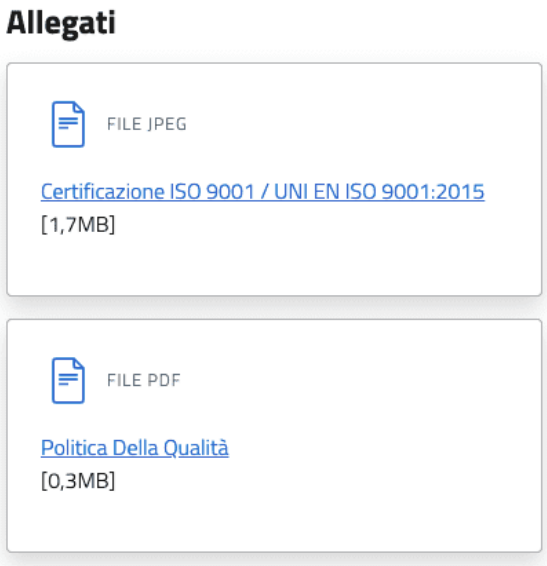
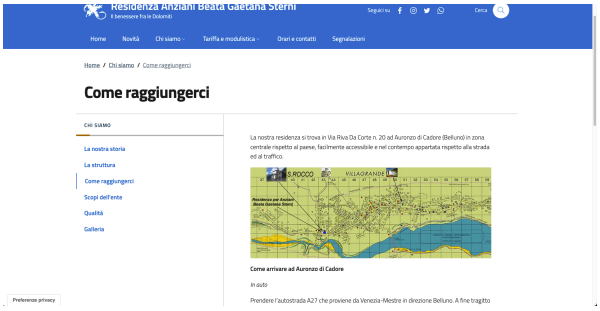


Figura 5.12: Esempio di allegati

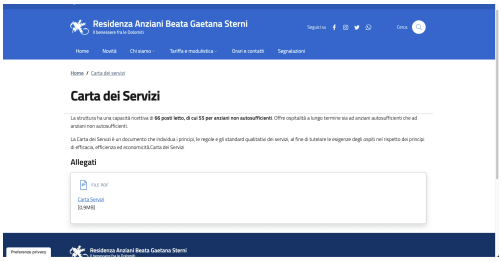
5.10 Standard Page

5.10.1 Aspetto

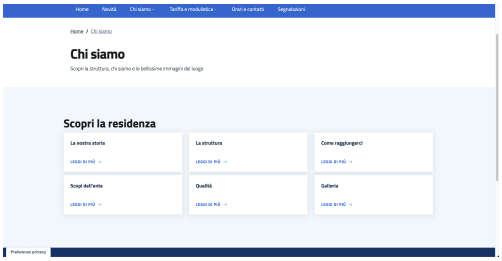


(a) Standard page da desktop

(b) Standard page da mobile



(c) Standard page con profondità < 2



(d) Standard Container

(e) Standard page da desktop

Figura 5.13: Standard Pages

Le **Standard Page** sono dei *Document Type* di base, utili per creare contenuti standard e non collegati ad un precisa precisa tassonomia. In questo modo si possono creare sezioni personalizzate nel sito, che funzioneranno allo stesso modo dei contenuti di tipo *Container*-Pagina; ovviamente a discapito di una resa visiva più dettagliata.
Gli **Standard Container** vengono ridotti all'essenziale, per permettere la maggior estendibilità possibile, e raccolgono semplicemente i nodi figli in delle *Card* che ne riprendono nome e link alla pagina (Figura 5.13d).

Le pagine **Standard** offrono una struttura simile a quelle già analizzate, ma tentano di generalizzare la struttura il più possibile (Figura 5.13a e 5.13e). Per elaborare il contenuto principale si utilizza un *Grid Layout*, che, come nella HomePage, permette di creare una vista personalizzata inserendo *RTE*, immagini, tabelle, link e molto altro: tutto disponendolo su un sistema a griglia, personalizzabile nel numero di colonne e righe (Figura 5.14a, 5.14b e 5.14c). In questo modo si offre una piena libertà nella gestione, ma naturalmente questo rende impossibile creare una struttura di *skiplink* nell'indice laterale. Per questo motivo la *NavBar* farà riferimento non più alla pagina in sè, bensì alla pagina padre; permettendo di indicizzarne le pagine figlie e quindi correlate a quella attualmente in osservazione.
Se una pagina standard è direttamente figlia della HomePage, il menu laterale non verrà renderizzato (Figura 5.13c). Da mobile il tutto si adatta verticalizzando il contenuto nella griglia e nascondendo il menu laterale in un link che alla pressione lo richiama (Figura 5.13b ed esempio di menu laterale simile a Figura 5.5c). Queste pagine standard offrono inoltre la possibilità di inserire allegati e caroselli di immagini.

5.10.2 Back End e proprietà

Le pagine *Container* presentano una struttura sempre uguale nel *Backend*, e quindi nel *Backoffice*. Questo permette di avere una struttura e una gestione dei contenuti sempre uniforme e coesa fra tutte le pagine dello stesso tipo. A cambiare sarà la presentazione nei nodi figli, gestita lato codice.
Le proprietà sono quindi le seguenti:

Standard Container

* **Titolo Pagina**: campo di testo con il titolo della pagina di tipo Container, che rispecchia la macro categoria di contenuti che raccoglie;
* **Descrizione Pagina**: area di testo con breve descrizione di cosa si tratta, presentata sotto il titolo;
* **Titolo sezione**: campo di testo che permette di inserire un titolo da dare alla sezione contenente la griglia di *Card*;
* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

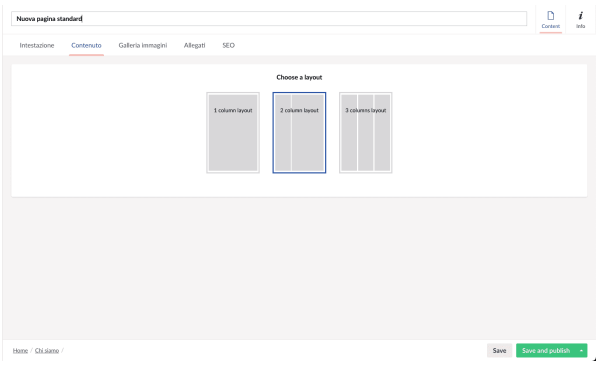
Standard Page

* **Titolo**: campo di testo con il titolo della pagina;
* **Contenuto**: *Grid Layout* che permette una gestione e un inserimento a piacere dei contenuti testuali e multimediali. Si può decidere se impostare un layout di partenza su una, due o tre colonne (Figura 5.14a) per poi suddividere e riempire

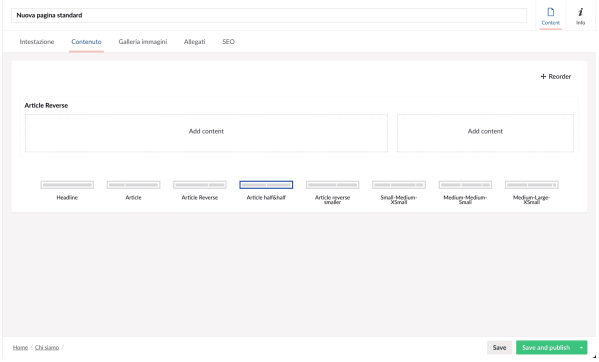
ogni riga a piacere (Figura 5.14b). Il rendering di Umbraco e un *Framework* personalizzato passato come parametro, chiamato `"bootstrap-italia"`, permettono una gestione automatica della griglia adattandola allo stile della libreria *Bootstrap Italia*;

* **Galleria immagini**: *Multiple Image Media Picker* che permette di selezionare una o più immagini da inserire in un carosello, avendo cura di inserire il testo alternativo;
* **Documenti Allegati**: *Multiple Media Picker* per inserire uno i più file allegati;
* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

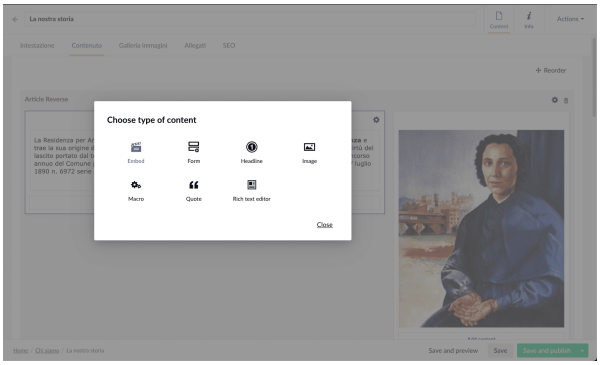
5.10.3 Dettagli implementativi



(a) Selezione layout di partenza



(b) Selettore di riga



(c) Selettore di *Data Type*

Figura 5.14: Creazione di contenuto Grid nel Backoffice

La particolarità di queste "*Standard Pages*" è il sistema di griglie.
Come detto, nel *Backoffice* di Umbraco, questo *Data Type* permette una potente e libera gestione dei contenuti, ottenendo così la massima flessibilità.
Nel *Frontend* si utilizza il metodo `GetGridHtml()` applicato al documento HTML. Pertanto, in questo caso, è stato utilizzato nel modo che segue al Listing 5.18.

```
<!-- Contenuto principale tramite grid layout -->
<article class="it-page-section anchor-offset">
    @Html.GetGridHtml(Model, "descrizione", "bootstrap-italia")
</article>
```

Listing 5.18: uso di GetGridHtml()

Si noti come il metodo accetti tre parametri (Listing 5.18), rispettivamente: il *Document Type* di riferimento, in questo caso quindi il `Model` in sè, l'*alias* della proprietà in cui trovare il contenuto e il *Framework* da utilizzare. Quest'ultimo definisce le regole per il rendering della griglia, permettendo di inserire le stesse classi utilizzate in *Bootstrap Italia*. Ad esempio il framework `boostrap-italia` qui creato, utilizza la seguente funzione per il rendering delle row (Listing 5.19).

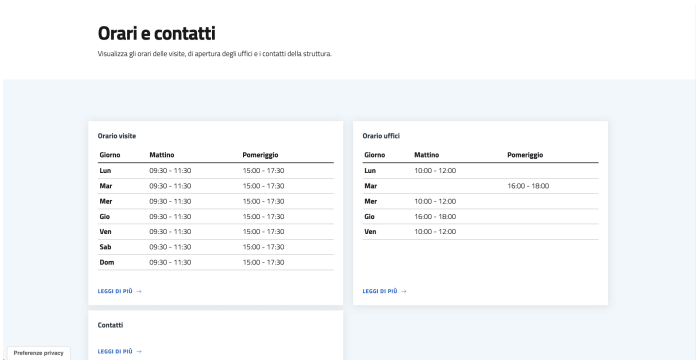
```
@functions{

private async Task renderRow(dynamic row, bool singleColumn)
{
<div @RenderElementAttributes(row)>
 @if (singleColumn) {
  @:<div class="container">
 }
 <div class="row clearfix">
  @foreach (var area in row.areas)
  {
   <div class="col-md-@area.grid column">
    <div @RenderElementAttributes(area)>
     @foreach (var control in area.controls)
     {
      if (control?.editor?.view != null)
      {
       <text>@await Html.PartialAsync("grid/editors/base",
            (object)control)</text>
      }
     }
    </div>
   </div>
   }
 </div>
 @if (singleColumn) {
  @:</div>
 }
</div>
}
}
```

Listing 5.19: Sistema di rendering delle row nel grid layout

Per far si che le "Standard Page" direttamente figlie del nodo *Root* non mostrassero il menu laterale, considerato poco utile, viene fatto un controllo dinamico sul livello della pagina. Viene quindi creata una variabile `var level = Model.Level;` che viene utilizzata in una condizione logica che gestisce il menu, e se questa variabile risulta maggiore di 2, il rendering non avviene.

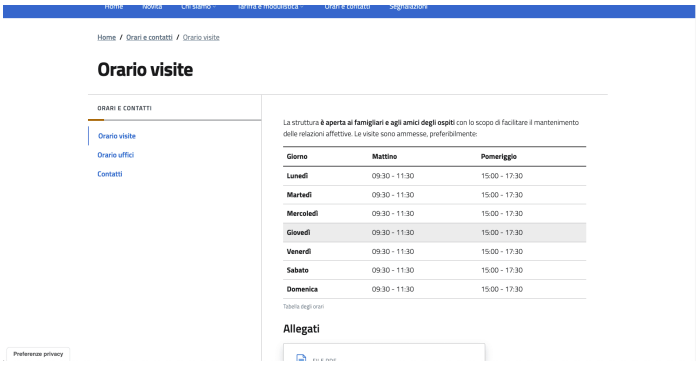
5.11 Orari e contatti

5.11.1 Aspetto



(a) Container Orari e Contatti

(b) Pagina orario da mobile



(c) Pagina orari da desktop

Figura 5.15: Pagine Orari e contatti

Le pagine di **Orari e Contatti** si presentano sostanzialmente come delle "Standard Page" con la principale aggiunta di una *composition*, nel *Backoffice*, per l'inserimento degli orari. Ad eccezione di questa aggiunta, rimangono sostanzialmente invariate nella presentazione rispetto a quanto visto nella sezione precedente.
La pagina padre rimuove il titolo e dispone le *Card* in un sistema di griglie (Figura 5.15a). Questi riquadri presentano subito una tabella degli orari, completamente accessibile, per far recepire subito l'informazione più importante all'utente appena entra nella sezione.
Rimane naturalmente la possibilità di entrare nella pagina dedicata per apprendere maggiori dettagli, se inseriti. La *Card* dei contatti è riportata in ultimo, in quanto le sue informazioni sono universalmente reperibili nel Footer del sito, nonostante possa presentare solamente i recapiti più importanti a discapito di quelli secondari.

Entrati in una pagina contente gli **Orari**, questi vengono presentati, anche qui, con

una tabella accessibile e con tutti i dettagli richiesti dalle linee guida (Figura 5.15c). Passando sopra una riga della tabella, inoltre, questa verrà evidenziata con un effetto di *hover*. Le tabelle scalano gradualmente con il ridursi dell'*Aspect Ratio* e anche da mobile (Figura 5.15b) si riesce a mantenerne invariata la struttura, garantendo un'adeguata leggibilità.

5.11.2 Back End e proprietà

Il *Backend* del *Container* di "Orari e Contatti" e le singole pagine presentano le seguenti proprietà:

orari e contatti Container

* **Titolo Pagina**: campo di testo con il titolo della pagina di tipo Container, che rispecchia la macro categoria di contenuti che raccoglie;
* **Descrizione Pagina**: area di testo con breve descrizione di cosa si tratta, presentata sotto il titolo;
* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

Orari e contatti

* **Titolo pagina**: campo di testo con il nome della pagina;
* **Contenuto**: *Grid Layout* che permette di inserire un contenuto aggiuntivo. Utile per l'inserimento dei contatti con libertà di presentazione;
* **Documenti Allegati**: *Multiple Media Picker* per inserire uno i più file allegati;
* **Orari**: *composition* contenente un *Nested content* che consente di scegliere gli orari giornalieri da mostrare nelle tabelle.
  Ogni elemento sarà quindi un *element* "Orari settimanali" che permette di scegliere un giorno della settimana e corrispettivi orari di apertura e chiusura (Figura 5.16). La visualizzazione corretta degli orari è poi gestita da codice Razor, così da renderli perfettamente leggibili anche per chi usa uno screen reader. Queste informazioni vengono utilizzate anche dalla pagina padre;
* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

5.11.3 Dettagli implementativi

L'aggiunta principale degna di nota in questi *Document Type* sono le tabelle, rese accessibili grazie ad una manipolazione dei dati a lato codice.
Come già visto per gli uffici, gli orari vengono inseriti in modo "grezzo" e per una tecnologia assistiva questo non va bene. Se non elaborati, il codice di Markup che ne risulterà verrà letto in forma di data anziché di ora, o in semplice numero.

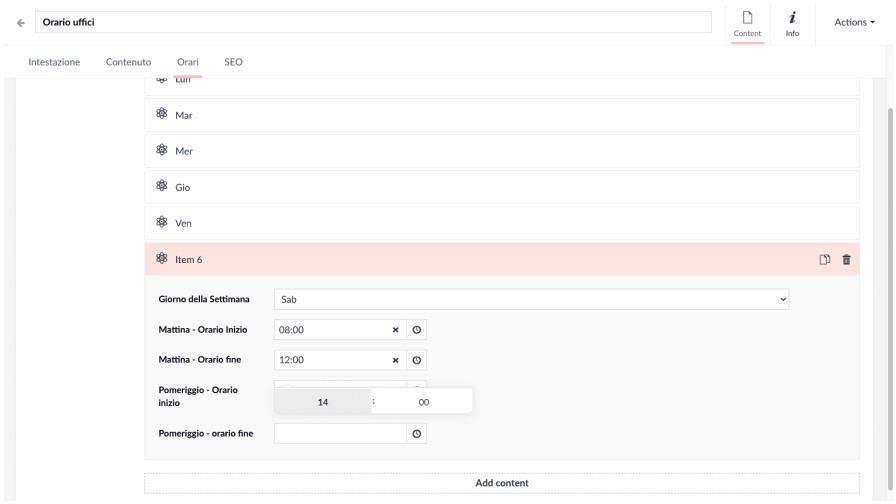


Figura 5.16: Backoffice per l'inserimento di un orario

Nella sezione che tratta gli uffici (sezione 5.7) si è gia visto il codice utile a rendere gli orari ricchi di informazione per gli screen reader (Listing 5.16). Di conseguenza, reimpiegando quel codice, una tabella viene gestita come segue al Listing 5.20.

```
@if (orari.Any()) {
<table class="table table-hover">
 <caption>Tabella degli orari</caption>
 <thead>
  <tr>
   <th scope="col">Giorno</th>
   <th scope="col">Mattino</th>
   <th scope="col">Pomeriggio</th>
  </tr>
 </thead>
 <tbody>
  @foreach (var item in orari) {
  string day = shortToLongDay(item.Value<string>("giornoDellaSettimana"));
  <tr>
   <th scope="row">@day</th>
   <td> @((item.Value<DateTime>("mattinaOrarioInizio").ToShortTimeString()) !=
        "00:00" ?
        item.Value<DateTime>("mattinaOrarioInizio").ToShortTimeString() + " -
        " + item.Value<DateTime>("mattinaOrarioFine").ToShortTimeString() :
        "") </td>
   <td> @((item.Value<DateTime>("PomeriggioOrarioInizio").ToShortTimeString())
        != "00:00" ?
        item.Value<DateTime>("PomeriggioOrarioInizio").ToShortTimeString() + "
        - " + item.Value<DateTime>("PomeriggioOrarioFine").ToShortTimeString()
        : "") </td>
  </tr>
[...]
}
```

Listing 5.20: Codice per una tabella accessibile

Si osservi come si possano utilizzare senza problemi gli `scope` della tabella, la `caption` e il tag `<th>` (Listing 5.20). Grazie al codice dinamico, in base al numero di giorni inseriti nel *nested content* verranno renderizzate il giusto quantitativo di righe con tanto di nome e `scope="row"`. Nella pagina padre (Figura 5.15a) viene riutilizzato lo stesso codice rimuovendo la classe `table-hover`.

5.12 Plugin esterni e Form

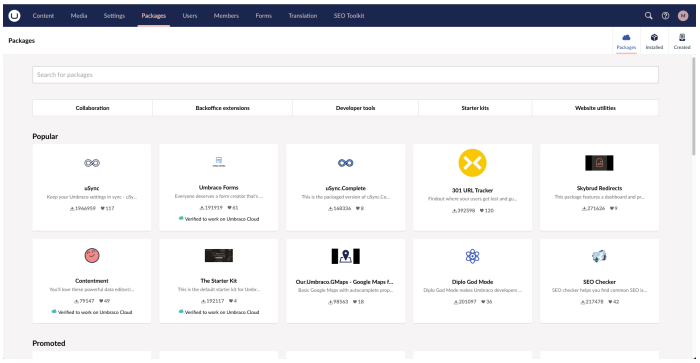


Figura 5.17: Tab per plugin esterni

Umbraco permette di estendere i propri contenuti tramite l'installazione di **Plugin** esterni.
Questi si possono installare sia da una apposita sezione interna al CMS (Figura 5.17) sia tramite installazione di pacchetti *NuGet*, metodo preferito durante sviluppo. All'interno del file `UmbracoCiv.csproj` si andranno quindi a creare dei riferimenti alle dipendenze. Questi *Plugin* permettono di aggiungere *Data Type*, nuove impostazioni, estensioni e *utility* varie.
in questo progetto sono stati installate tre estensioni:

* **Contentment**: aggiunge svariati *Data Type* non previsti da Umbraco, primo fra tutti il *Data List*, utilizzato svariate volte per creare un picker a nodi, o altri contenuti, non direttamente collegati al *Document Type* in gestione;

* **SEO Toolkit**: permette di creare in automatico i file `sitemap.xml` e `robots.txt`, che si aggiorneranno in automatico in base alle modifiche. Si può inoltre estendere il plugin permettendogli di creare i *redirect* automatici delle pagine e i *meta field*, in modo simile a quanto fatto manualmente con la *composition* "Seo". Entrambi vengono automaticamente resi indicizzabili nella sito e si possono vedere andando al percorso `.../sitemap.xml` o `.../robots.txt`;

* **Umbraco Forms**: permette di creare *Form* personalizzati, direttamente da una facile interfaccia visuale nel *Backoffice*. Una volta creato un form gli si possono applicare uno stile personalizzato e inserirlo dove si preferisce nel sito. Si potrà poi anche avere a disposizione un record delle compilazioni avvenute e modificare il *workflow* che segue all'invio dei dati.

5.12.1 Approfondimento sui Forms

L'estensione **Umbraco Forms** si presenta in una Tab apposita all'interno del CMS. Una volta entrati nel menu, si possono modificare e creare dei nuovi Form o controllare i record di quelli presenti (Figura 5.18).

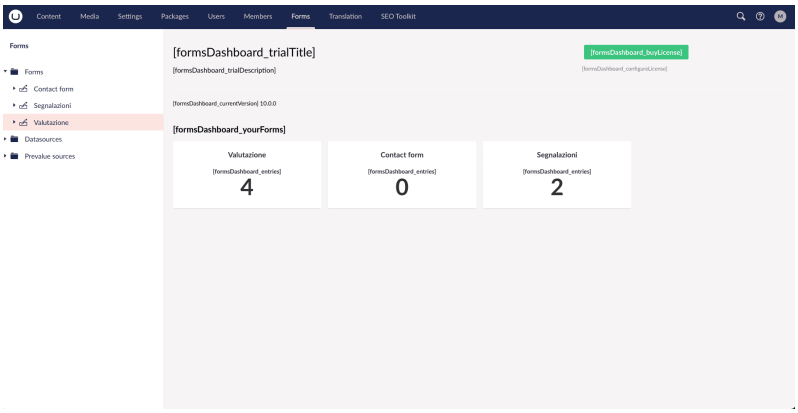
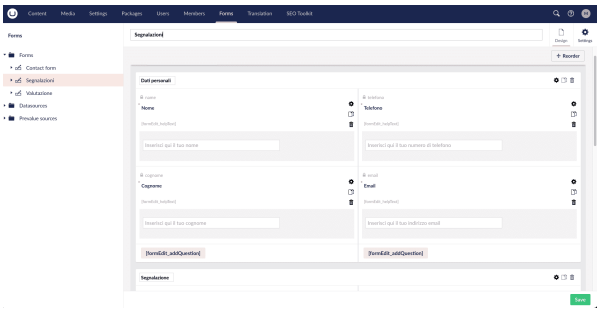
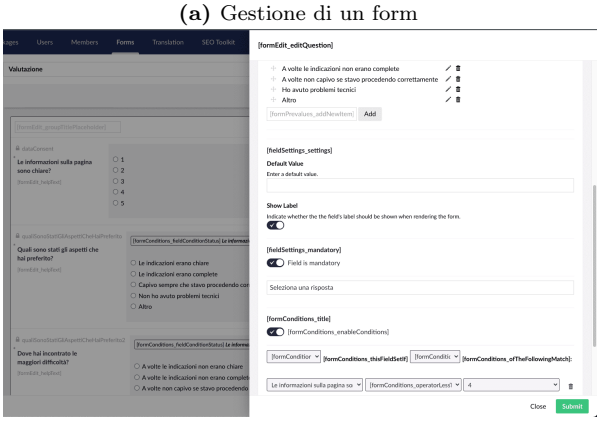


Figura 5.18: Tab Forms

Nella pagina di gestione (Figura 5.19a) possiamo inserire nuovi campi associandogli sempre un *Data Type* per poi valorizzare le proprietà con vari attributi come: l'obbligatorietà di compilazione, il *placeholder*, la label, l'autocompilazione e condizioni logiche (Figura 5.19b).



(a) Gestione di un form



(b) Gestione di una proprietà

Figura 5.19: Pagine Orari e contatti

Le condizioni logiche permettono di definire, ad esempio, se far apparire un input solo ad una determinata condizione dettata dal valore inserito in un campo precedente.

Quest'ultima opzione è utile ad esempio nelle pagine di valutazione (Figura 5.20, destra), dove in base al valore dato (in una scala da 1 a 5), apparirà di conseguenza un questionario differente o un'area di testo (se si seleziona la classica opzione "altro"). Umbraco Forms permette inoltre una gestione automatica degli errori, spostando il focus sugli input errati, le cui validazioni si possono fare tramite *RegEx*, e con la possibilità di inserire un messaggio personalizzato.

Una volta disposte le varie sezioni a piacimento, il codice inserito nel *template* si occuperà del resto. Il *Frontend* segue al Listing 5.21.

```
<div class="row">
  <div class="col">
   @await Umbraco.RenderMacroAsync("renderUmbracoForm", new
        {FormGuid=Model.Value("form"), FormTheme="bootstrap-italia"})
  </div>
</div>
```

Listing 5.21: Rendering di un form

Come per i *Grid Layout*, anche qui si può decidere di personalizzare il rendering del contenuto, assegnando al parametro `FormTheme` il *Framework* desiderato. In questo caso si utilizza *bootstrap-italia* per permetterne la visualizzazione come definita nella libreria grafica.
I risultati ottenuti saranno simili ai seguenti, mostrati alla Figura 5.20.

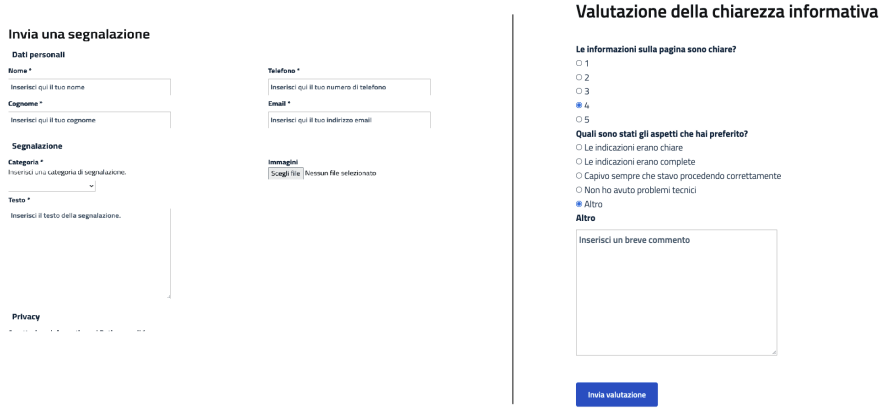


Figura 5.20: Form segnalazioni e valutazione

Questa estensione, come le altre, è di fatto una dipendenza dichiarata nel file `UmbracoCiv.csproj` (Listing 5.22). Per aggiornare quindi la componente sarà necessario modificare manualmente il numero di versione per poi fare una rebuild del progetto, oppure lasciare alla gestione automatica dei pacchetti NuGet, in Visual Studio, tutto il lavoro.

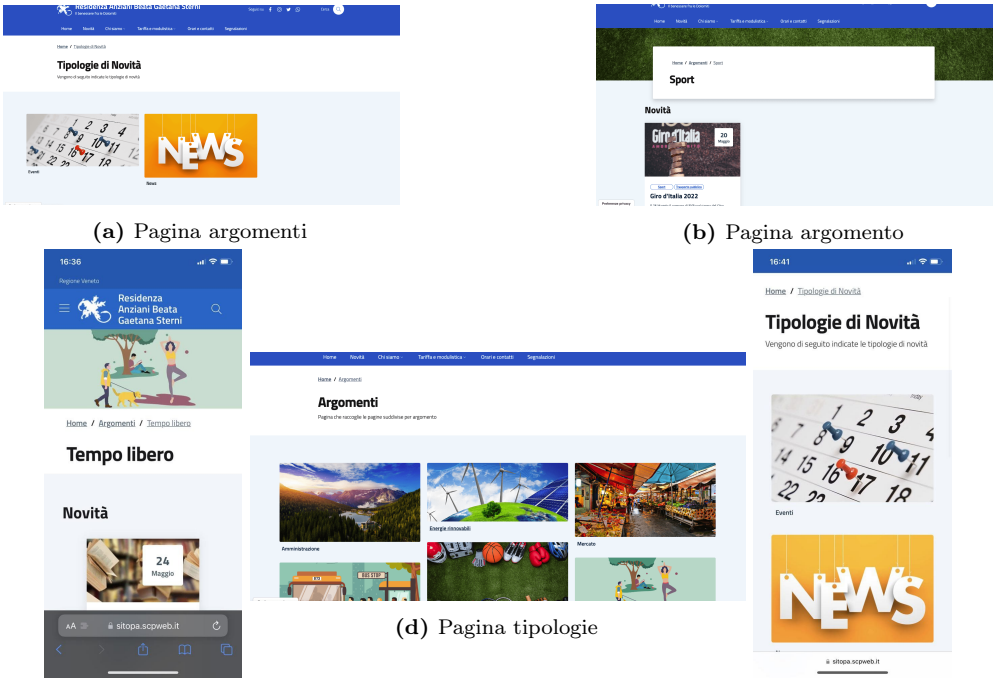
```
<ItemGroup>
 [...]
 <PackageReference Include="Umbraco.Forms" Version="10.0.1" />
</ItemGroup>
```

Listing 5.22: Dipendenze nel file `UmbracoCiv.csproj`

5.13 Argomenti, Tipologie e Categorie

Nelle pagine di tipo "Notizia" e "Servizio" si è visto come ci siamo delle *Chip* che indicano l'argomento, la tipologia o la categoria di appartenenza (Figura 5.7c e 5.10c). Questi elementi possono essere cliccati e il loro reindirizzamento porterà a dalle pagine che raccolgono tutti i contenuti affini.

5.13.1 Aspetto



(a) Pagina argomenti

(b) Pagina argomento

(c) Pagina argomento da mobile

(d) Pagina tipologie

(e) Pagina tipologie da mobile

Figura 5.21: Argomenti, Tipologie e Categorie

Queste pagine seguono la classica gerarchia di contenuti, ma con uno stile sostanzialmente differente. Le pagine container utilizzano la libreria *Masonry*, già contenuta in *Bootstrap*, per mostrare le singole sezioni con una immagine che le rappresenti. Questa libreria permette di creare un effetto dinamico e fluido, garantendo una piacevole presentazione delle immagini che andranno a ridisporsi in modo animato con il resize della pagina.
Si noti come ogni argomento, tipologia e categoria deve essere un nodo pubblicato affinché sia indicizzabile dalle pagine che vi fanno riferimento. Le pagine container (Figura 5.21a, 5.21d e 5.21e) devono pertanto gestire i nodi figli mostrando solo quelli che effettivamente contengono qualcosa.

Le singole pagine hanno una intestazione composta da una *Hero* con l'immagine che le raffigura (o una di default in caso non sia inserita) e una *Card* orizzontale riportante breadcrumb e titolo (Figura 5.21b e 5.21c). Al di sotto si ritrovano, suddivisi

per sezioni, le "Novità" o i "Servizi" collegati, tramite un carosello. È prevista inoltre una sezione aggiuntiva per estensioni future, con *Card* meno elaborate.
Da mobile le pagine scalano verticalizzando il contenuto in modo gradevole e consistente (Figura 5.21c e 5.21e).

5.13.2 Back End e proprietà

Di seguito elencate le, poche, proprietà che caratterizzano e accomunano il *Backend* di argomenti, tipologie e categorie. Si precisa che i tre gruppi di contenuto sono tre differenti *Document Type*: questo è stato reso necessario per via delle differenze che, invece, presentano nel *Frontend*.

Argomenti, Tipologie e Categorie Container

* **Titolo Pagina**: campo di testo con il titolo della pagina di tipo Container, che rispecchia la macro categoria di contenuti che raccoglie;
* **Descrizione Pagina**: area di testo con breve descrizione di cosa si tratta, presentata sotto il titolo;
* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

Argomento, Tipologia e Categoria

* **Immagine**: *Image Media Picker* che permette l'inserimento di una immagine rappresentativa dell'argomento, tipologia o categoria. Se non inserita, verrà utilizzata un'immagine di default. Questo dato verrà utilizzato anche dalla pagina padre per creare la lista di immagini con *Masonry*.

5.13.3 Dettagli implementativi

Per permettere una più chiara gestione nel *Backoffice* e per un maggiore dettaglio lato codice e *Frontend*, queste pagine sono tutti dei *Document Type* differenti. Nonostante questo, buona parte del codice risulta simile se non uguale a meno di particolari caratteristiche legate alla natura delle pagina in sé, come i testi alternativi. In ogni pagina *container* viene utilizzato *Examine*[3], analizzato in seguito, per effettuare una ricerca in tutto il dominio.
Questo serve per cercare e filtrare i nodi figli solo a quelli che effettivamente sono attualmente riferiti nel sito: se quindi, per esempio, un determinato argomento (pur esistendo come nodo) non è riferito in alcuna notizia o servizio, questo non dovrà apparire nella lista del nodo padre, perché indicizzerebbe ad una pagina di contenuto vuoto. Ad inizializzazione della pagina viene quindi effettuata una *query* (Listing 5.23) su tutta la gerarchia di contenuti e in base al risultato ottenuto si mostreranno o meno i nodi figli nella lista di immagini con *Masonry* (es. Figura 5.21a e 5.21d).

[3] *Examine*. URL: https://shazwazza.github.io/Examine/.

```
@inject IExamineManager ExamineManager;
@using Examine;
@using System.Text.Json
[...]
@{
if (!ExamineManager.TryGetIndex(Constants.UmbracoIndexes.ExternalIndexName,
    out var index))
{
 throw new InvalidOperationException($"No index found by
     name{Constants.UmbracoIndexes.ExternalIndexName}");
}
Layout = "Master.cshtml";
var query = index.Searcher.CreateQuery().NativeQuery("categorie:[\"\" TO *]");
var results = query.Execute().SelectMany(t =>
    t.GetValues("categorie")).SelectMany(t =>
    JsonSerializer.Deserialize<string[]>(t)).Distinct().Select(t =>
    Umbraco.Content(t));
}
```

Listing 5.23: Filtraggio dei nodi figli con Examine

Come si può vedere nel caso delle "categorie" (Listing 5.23), una volta effettuata la query il risultato viene filtrato, per poi essere utilizzato come condizione logica nel rendering della lista, come segue al Listing 5.24.

```
@if (results != null)
{
<div class="section section-muted">
 <div class="section-content">
  <div class="container-fluid">
   <div class="row it-masonry" data-bs-toggle="masonry">
    @foreach (var item in results.OrderBy(x => x!.Name))
    {
    <div class="col-sm-6 col-lg-4 mb-4">
     <div class="it-grid-item-wrapper">
      <a href="@item.Url()">
       <div class="img-responsive-wrapper">
        <div class="img-responsive">
         @{var argImg = item.Value<MediaWithCrops>("immagine");}
         <div class="img-wrapper"><img
          src="@( argImg != null ? argImg.MediaUrl() :
              Umbraco.Media("1310").MediaUrl())"
          alt="Foto dell'argomento" aria-hidden="true"/>
         </div>
        </div>
       </div>
      <span class="it-griditem-text-wrapper">
      <span class="it-griditem-text">@item.Name</span>
    [...]
```

Listing 5.24: Rendering lista di immagini con Masonry

Le classi che definiscono il layout di base sono quindi definite dalla libreria (ad esempio *it-masonry, it-grid-item-wrapper*) ma il contenuto della griglia è completamente

manipolabile in base alle esigenze. In questo caso specifico non è stato però possibile converite le immagini in formato *WebP*[g] (tramite il plugin *ImageSharp*), come nel resto del sito, per via di un errore che si genera negli operatori *ternary*.

All'interno di ogni singola pagina poi, viene riutilizzato *Examine* per trovare tutti i nodi che associano a quell'argomento, tipologia o categoria.
Qui il processo di ricerca (Listing 5.25) si esegue in modo differente: come prima cosa viene generata una variabile `term` che sarà il nome della pagina in sè convertito in stringa. Sucessivamente verrà eseguita una *query* per cercare tutti i nodi del sito che hanno, nelle loro proprietà, `term` nel campo apposito. Ad esempio, se la variabile di ricerca è "Sport", il metodo `.ManagedQuery()` tornerà tutte le pagine che hanno quella stringa come argomento.
Ottenuto il risultato, gli elementi trovati verranno filtrati in base al *Document Type*.

```
if (!ExamineManager.TryGetIndex(Constants.UmbracoIndexes.ExternalIndexName,
    out var index))
{
 throw new InvalidOperationException($"No index found by name{
     Constants.UmbracoIndexes.ExternalIndexName }");
}
var term = Udi.Create(Constants.UdiEntityType.Document, Model.Key).ToString();
var query = index.Searcher.CreateQuery(IndexTypes.Content);
var queryExecutor = query.ManagedQuery(term, new string[] {"categorie"});
var results = queryExecutor.Execute().GroupBy(t =>
    t.Values["__NodeTypeAlias"]);
```

Listing 5.25: Ricerca dei nodi associati al tipo di pagina

Per il rendering si utilizza la stessa procedura di creazione di caroselli a scorrimento orizzontale, accedendo ai dati di ogni singolo nodo tramite l'`Id` (Listing 5.26).

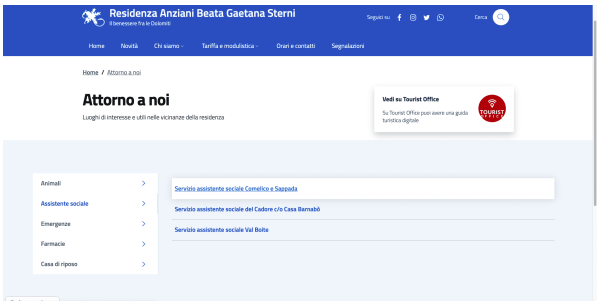
```
@foreach (var node in group.Select(t =>
    Umbraco.Content(t.Id)).OrderByDescending(t => t.CreateDate))}
```

Listing 5.26: `foreach` sui nodi trovati

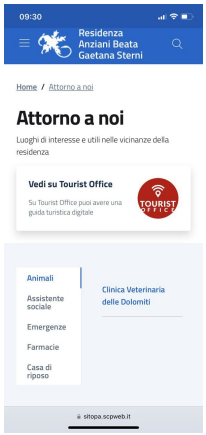
Laddove, attualmente, ad un argomento si riferiscono sia novità che servizi, e pertanto vi deve essere una distinzione dei due tipi in due sezioni differenti; ad una tipologia e ad una categoria vi si riferiscono rispettivamente solo uno o l'altro. Come già accennato, però, in caso un risultato non appartenga a nessuna dei due tipi, verrà renderizzata una sezione con *Card* basilari composte da nome e link.

5.14 Luoghi di interesse

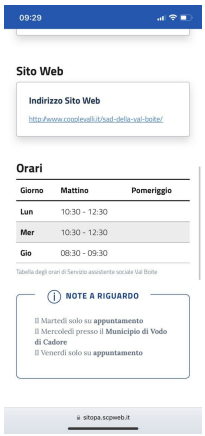
5.14.1 Aspetto



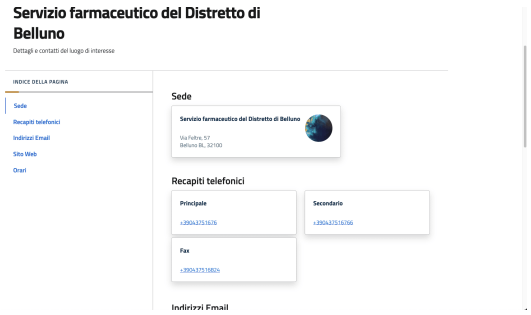
(a) Pagina luoghi di interesse da desktop



(b) Pagina luoghi di interesse da mobile



(c) Luogo da mobile



(d) Luogo da desktop

Figura 5.22: Luoghi di interesse e singolo luogo

Utili per segnalare i **luoghi di interesse**, queste pagine seguono la già vista gerarchia di contenuti, presentando questa volta i nodi figli in un menu *Tab* (Figura 5.22a e 5.22b). Questo permette di aggiungere in modo compatto e intuitivo i nuovi luoghi man mano che essi verranno aggiunti, suddividendoli in automatico in base alla categoria di appartenenza. Ogni voce della *Tab* viene generata dinamicamente così come la lista che contiene.
Ogni singolo **Luogo** mostra i dati relativi alla sede, i contatti e gli orari, nonché un eventuale *Callout* per note aggiuntive (Figura 5.22c). Naturalmente rimane la possibilità di inserire recapiti multipli (Figura 5.22d) e la tabella degli orari segue tutte le indicazioni di accessibilità.

In testa alla pagina *container* è stato inoltre inserito un link rapido, modificabile o rimovibile, ad un servizio creato e gestito dall'azienda ospitante.

5.14.2 Back End e proprietà

Il *Backend* presenta le seguenti proprietà:

Luoghi di interesse Container

* **Titolo Pagina**: campo di testo con il titolo della pagina di tipo Container, che rispecchia la macro categoria di contenuti che raccoglie;
* **Descrizione Pagina**: area di testo con breve descrizione di cosa si tratta, presentata sotto il titolo;
* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

Luogo

* **Nome luogo**: campo di testo con il nome del luogo;
* **Categoria Luogo**: *Dropdown Menu* che permette di scegliere, fra una lista di predefiniti, la categoria di appartenenza del luogo;
* **Descrizione**: *RTE* per inserire una breve descrizione;
* **Numeri di telefono**: *Nested content* che permette di scegliere i numeri di telefono da mostrare nella sezione dei contatti.
  Ogni elemento sarà quindi un *element* "Telefono";
* **Emails**: *Nested content* che permette di scegliere le email da mostrare nella sezione dei contatti.
  Ogni elemento sarà quindi un *element* "Email", cui funzionamento è analogo a quello dei numeri di telefono;
* **Sito Web**: selettore di URL per inserire un link al sito web del luogo;
* **Indirizzo**: *composition* che permette di inserire il nome della via, il numero civico, il nome del Comune, sigla della Provincia e CAP della Sede;
* **Orari**: *composition* contenente un *Nested content* che permette di scegliere gli orari giornalieri da mostrare nella tabella dedicata. Ogni elemento sarà quindi un *element* "Orari settimanali";
* **Note aggiuntive**: *RTE* per inserire note aggiuntive;
* **SEO**: *composition* che permette di inserire il *title* e la *description* della pagina.

5.14.3 Dettagli implementativi

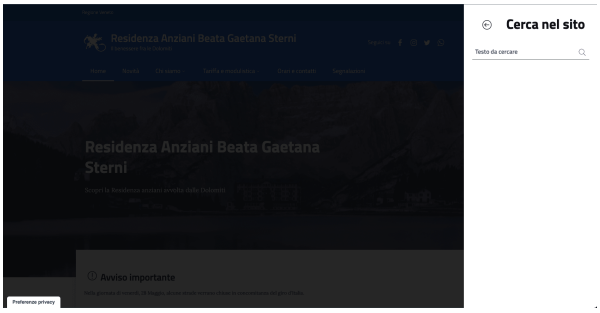
Queste pagine seguono una struttura affine a quelle già portate in analisi, utilizzando componenti anch'esse analoghe e gestite allo stesso modo in termini di accessibilità e rendering. Si differenzia solamente la *Tab* presente nella pagina padre, costruibile nel *Frontend* con l'esempio di codice che segue (Listing 5.27).

```
<div class="bd-example-tabs">
 <div class="row">
  <div class="col-5 col-md-4 col-lg-3">
   <div class="nav nav-tabs nav-tabs-vertical" id="nav-vertical-tab-ico"
       role="tablist" aria-orientation="vertical">
    @{ int i = 0; }
    @foreach (var item in luoghi)
    {
    i += 1;
    <a class="nav-link @( i == 1 ? " active" : string.Empty)"
        id="nav-vertical-tab@(i)-tab" data-bs-toggle="tab"
        href="#nav-vertical-tab@(i)" role="tab"
        aria-controls="nav-vertical-tab@(i)" aria-selected="true">
     @item.Key
     <svg class="icon icon-primary d-none d-md-block">
         <use href="/assets/bootstrap-italia/svg/..."></use>
     </svg>
    </a>
    }
   </div>
  </div>
   <div class="col-7 col-md-8 col-lg-9">
    <div class="tab-content" id="nav-vertical-tab-icoContent">
    @{int j = 0;}
    @foreach (var item in luoghi)
    {
    [...]
```

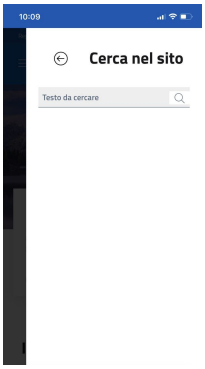
Listing 5.27: Tab di contenuti

5.15 Ricerca globale con Examine

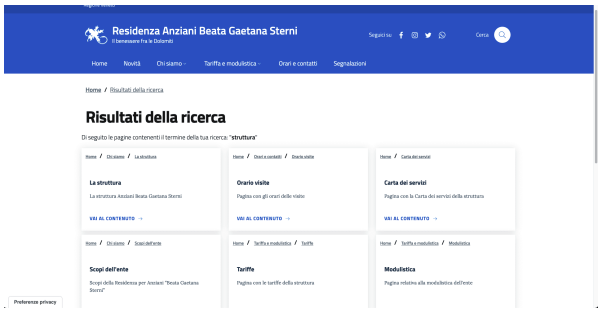
5.15.1 Aspetto



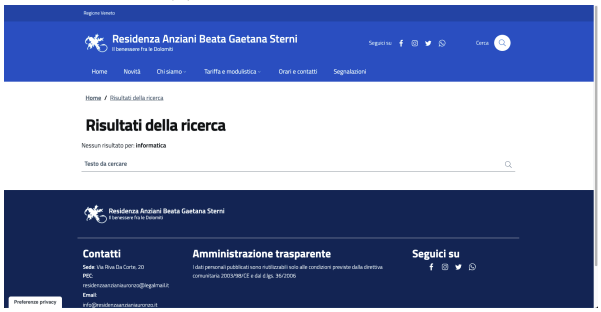
(a) Ricerca da desktop



(b) Ricerca da mobile



(c) Risultati di una ricerca



(d) Risultato di ricerca nullo

Figura 5.23: Ricerca nel sito

Sono già stati portati in analisi degli esempi di ricerca tramite la libreria *Examine* già integrata in Umbraco. Questa permette di eseguire, in modo agile e rapido, delle ricerche globali sull'intero dominio basandosi su un array di *alias* cui riferirsi (Listing 5.30). Si può inoltre decidere di eseguire questa operazione solo su indici interni, o solo esterni o entrambi, filtrando quindi tra i nodi pubblicati e non. Eseguendo la *query* si ottiene un risultato, manipolabile a piacere, e in base a questo si può creare un rendering dinamico e condizionale. Ad esempio, se il termine non viene trovato o è nullo, verrà

segnalato tramite un messaggio e verrà riproposto un campo di ricerca (Figura 5.23d). La schermata di ricerca viene richiamata tramite una modale (Figura 5.23a e 5.23b), con un input che ottiene immediatamente il *focus*. Ottenuti i risultati, verranno mostrate delle *Card* con il nome della pagina che contiene il termine, un link rapido e una breadcrumb (Figura 5.23c).

5.15.2 Dettagli implementativi

La modale di ricerca conterrà un classico `<form>` con tag `<input>` (Listing 5.28).

```
<div class="modal-body">
 <form action="@searchPage!.Url()" method="get">
  <div class="form-group">
   <label for="search" class="visually-hidden">Cerca nel sito</label>
   <input type="search" class="autocomplete" placeholder="Testo da cercare"
       id="search" name="query"
    />
   <button type="submit" class="autocomplete-icon" aria-hidden="true"
       style="border: none;" aria-label="avvia ricerca">
    <svg class="icon icon-sm"><use
        href="/assets/bootstrap-italia/svg/..."></use></svg>
   </button>
  </div>
 </form>
</div>
```

Listing 5.28: Form nella modale di ricerca

Il punto focale è la `action` della form, che rimanda all'URL della variabile `var searchPage = Umbraco.ContentAtXPath("//searchResults").FirstOrDefault();`, cioè la pagina dove vengono mostrati i risultati di ricerca. È in quest'ultima che viene eseguita la *query*, passandole in entrata il parametro del form. Viene quindi processato il dato in entrata, facendo i dovuti controlli e operazioni di *Trim* sul testo cercato, come visibile al Listing 5.29.

```
var searchPage = Umbraco.ContentAtXPath("//searchResults").FirstOrDefault();
var searchTerm = string.Empty;
searchTerm =
 string.IsNullOrEmpty(HttpContextAccessor!.HttpContext!.Request.Query["query"][0])
 ? string.Empty
 : HttpContextAccessor.HttpContext.Request.Query["query"][0].Trim();
```

Listing 5.29: Analisi dell'input

Ora, se la variabile rimane una stringa vuota verrà riproposto un form di ricerca con un messaggio che avvisa l'utente del fallimento dell'operazione (Figura 5.23d), altrimenti verrà eseguita la *query* come segue al Listing 5.30.

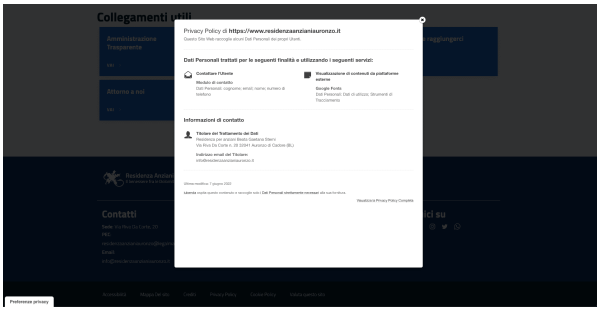
```
var searcher = index.Searcher;
 var textFields = new List<string> { "nodeName", "titolo", "sottotitolo",
     "descrizione", "competenze", "comune", "persona", "categoria", "tipo",
     "argomento", "description", "content" };
 var results = searcher.CreateQuery("content").GroupedOr(textFields,
     searchTerm).Execute();
 if (results.Any())
 {
     [...]
 }
```

Listing 5.30: Query

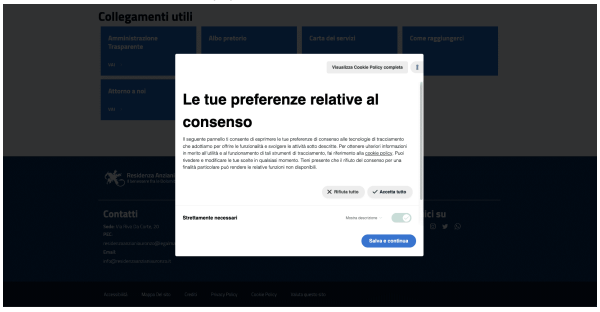
Anche qui, se non viene trovato nessun risultato verrà visualizzata una schermata analoga alla ricerca vuota. Fondamentale è la lista `textFields` (Listing 5.30), che contiene gli *alias* cui riferirsi per la ricerca.
Risulta chiaro che generalizzando il *Backend* in modo adeguato, riutilizzando gli stessi *alias* per proprietà analoghe, si può ottimizzare in modo significativo il processo e aumentarne la manutenibilità.

5.16 Privacy

Per la gestione della *Privacy e Cookie Policy* ci si è affidati ad uno strumento partner dell'azienda ospitante, ovvero **Iubenda**[4].
Questo strumento online permette, una volta inseriti il dominio e i dati della proprietà, di creare in automatico delle dichiarazioni sulla privacy analizzando i cookie instanziati nel sito.
Di conseguenza permette di creare i famosi *PopUp* per il consenso del trattamento dei dati, da inserire come `<script>` nella pagina.
Allo stesso modo le informative possono essere inserite, solitamente nel Footer della pagina, come link; quali possono aprire un comodo *PopUp* per avere maggiori dettagli (Figura 5.24a e 5.24b).



(a) Privacy Policy



(b) Consenso dei Cookies

Figura 5.24: PopUps delle informative sulla privacy

Si noti bene che qualsiasi strumento che possa raccogliere dei dati dell'utente deve essere dichiarato nella *Privacy Policy* come definito nel *GDPR*. Pertanto anche un form che contenga la manipolazione di dati sensibili (nome, cognome, numero di telefono ecc.) deve essere correttamente reso noto nella informativa, la quale deve essere sempre presente.

[4] *Iubenda*. URL: https://www.iubenda.com/it/.

5.17 Pubblicazione

Infine, la pubblicazione del sito deve avvenire tramite un file di configurazione (Listing 5.31) generato automaticamente, e solamente, da *Visual Studio*.

```
<?xml version="1.0" encoding="utf-8"?>
<Project>
    <PropertyGroup>
    <WebPublishMethod>MSDeploy</WebPublishMethod>
    <LastUsedBuildConfiguration>Release</LastUsedBuildConfiguration>
    <LastUsedPlatform>Any CPU</LastUsedPlatform>
    <SiteUrlToLaunchAfterPublish>http://miosito.it:80/</SiteUrlToLaunchAfter...>
    <LaunchSiteAfterPublish>true</LaunchSiteAfterPublish>
    <ExcludeApp_Data>false</ExcludeApp_Data>
    <ProjectGuid>[...]</ProjectGuid>
    <MSDeployServiceURL>https://S-FW01-WEB1:8172/msdeploy.axd</MSDeploy...>
    <DeployIisAppPath>[...]</DeployIisAppPath>
    <RemoteSitePhysicalPath />
    <SkipExtraFilesOnServer>true</SkipExtraFilesOnServer>
    <MSDeployPublishMethod>WMSVC</MSDeployPublishMethod>
    <EnableMSDeployBackup>true</EnableMSDeployBackup>
    <EnableMsDeployAppOffline>true</EnableMsDeployAppOffline>
    <UserName>[...]</UserName>
    <_SavePWD>true</_SavePWD>
    <TargetFramework>net6.0</TargetFramework>
    <SelfContained>false</SelfContained>
    </PropertyGroup>
</Project>
```

Listing 5.31: File di configurazione per la pubblicazione

Non è stato possibile eseguire questo processo dalla stessa macchina in cui veniva gestito il progetto, in quanto la funzionalità di "*Publish*" in Visual Studio non è supportata nella sua versione per Mac OS. Ci si è quindi affidati di volta in volta ad un tecnico che disponesse di tale funzione.

Capitolo 6

Test e accessibilità

Si analizzano, in questo capitolo, gli strumenti utilizzati a fini di testing e valutazione del prodotto

6.1 Premessa

Al fine di adeguarsi sia alle norme delle linee guida della Pubblica Amministrazione sia per ottenere uno standard qualitativo di alto livello, molti strumenti sono stati utilizzati in via di sviluppo per assicurarsi una continua verifica e validazione del codice.
In ottica di un approccio "*Accessible By Default*", un requisito fondamentale è stato quello di raggiungere le massime percentuali in termini di accessibilità, vista la natura del prodotto, cercando di superare i limiti che spesso ha imposto l'uso della libreria *Bootstrap Italia*, nonostante nasca a questo fine.
Altro punto importante è stato quello di garantire massima fruibilità da qualsiasi dispositivo, sia in termini di prestazioni che in termini di dimensioni dello schermo.
Alcuni strumenti sono stati individuati grazie alla piattaforma *MyWCAG4All*[1], creata da uno studente dell'Ateneo.
Nelle sottosezioni che seguono vengono quindi elencati i principali strumenti utilizzati e i riscontri ottenuti.

6.2 WAVE: Web Accessibility Evaluation Tool

Durante l'intero processo di sviluppo, è stata utilizzata l'estensione **WAVE**[2] per verificare la presenza di errori o meno nelle varie pagine create. Questo strumento permette di controllare fattori come:

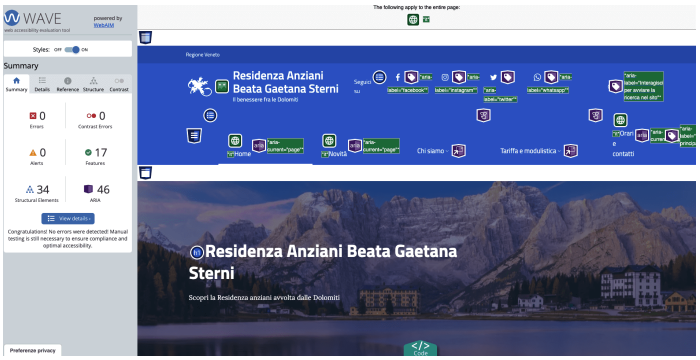
* La presenza di errori di accessibilità nel codice *HTML* prodotto;
* La presenza di errori nei contrasti;
* Warning di possibili problemi di accessibilità;
* Features implementate nel codice per favorire l' accessibilità;
* La strutturazione degli elementi della singola pagina;

[1] *MyWCAG4All*. URL: https://web.math.unipd.it/accessibility-dev/.
[2] *Estensione WAVE*. URL: https://wave.webaim.org/extension/.

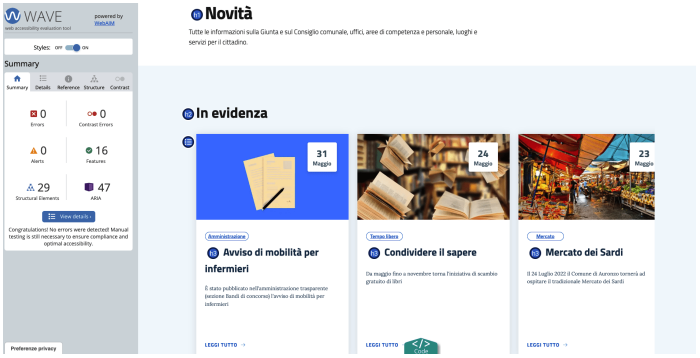
* Gli attributi *ARIA*.

L'uso continuo fin dall'inizio ha permesso di correggere man mano tutte le problematiche che venivano sollevate dallo strumento, assicurando la massima qualità del prodotto. A lavoro finito, tutte le pagine presentano un buon numero di features, di attributi *ARIA* e nessun errore di accessibilità.
Naturalmente non si può controllare il modo in cui verranno poi gestiti i contenuti dall'amministratore finale del sito. Grazie alla flessibilità offerta da Umbraco, però, è stato possibile inserire alcuni accorgimenti atti a correggere potenziali dimenticanze da parte di chi avrà poi in mano il prodotto; alcuni di questi sono visionabili nella sezione dedicata allo sviluppo.
Alcune criticità sono state riscontrate nelle prime versioni del prodotto che utilizzano una versione di *Bootstrap Italia* basata su *Bootstrap 4.6*[3], il quale non permetteva di cambiare il tag dei titoli senza perdere lo stile della classe. Naturalmente questo risultava un problema se lo stile di un intero componente era fortemente legato a quel determinato tag, in quanto spesso la struttura delle intestazioni non seguiva la corretta sequenza logica, generando degli errori. Eseguito l'upgrade a *Bootstrap Italia 2.0* però, questo problema è stato risolto ed è stato possibile assegnare i corretti livelli di intestazione. Nelle immagini che seguono vengono riportati alcuni esempi di applicazione dello strumento a prodotto finito (Figura 6.1a, 6.1b, 6.1c e 6.1d)

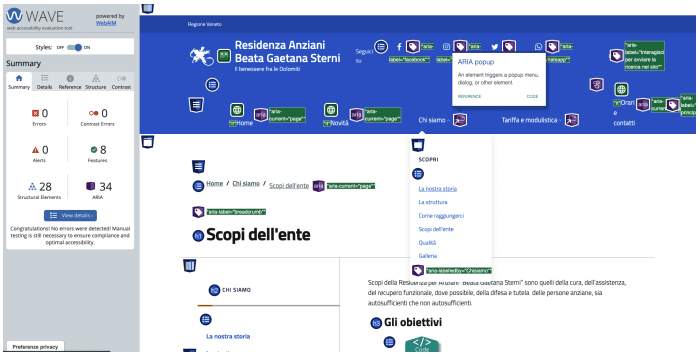
[3] *Bootstrap 4.6*. URL: https://getbootstrap.com/docs/4.6/getting-started/introduction/.



(a) Sulla HomePage



(b) Sulla pagina novità



(c) Su una standard page

(d) Su una pagina orari con tabella

Figura 6.1: Alcuni esiti dati dallo strumento WAVE

6.3 Lighthouse

Lo strumento **Lighthouse** è integrato nel browser **Chrome** e permette di eseguire una approfondita analisi della pagina che si sta attualmente visitando.
Questa estensione permette di verificare quattro fondamentali parametri, quali:

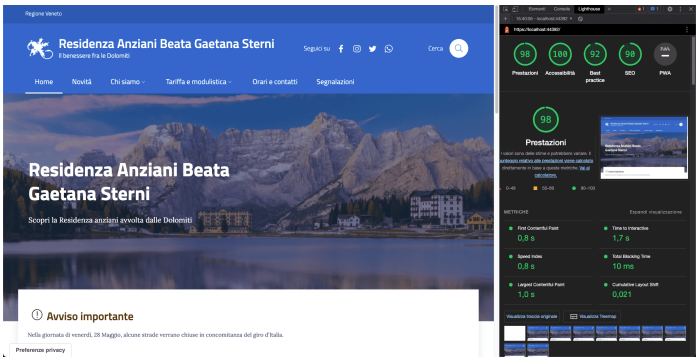
* Prestazioni;
* Accessibilità;
* Best practice;
* SEO.

In accoppiata con WAVE, quindi, ha permesso di eseguire un doppio controllo sull'accessibilità e la correttezza del codice, nonché delle verifiche sulle performance.
Per ogni pagina si è puntato ad avere i seguenti punteggi:

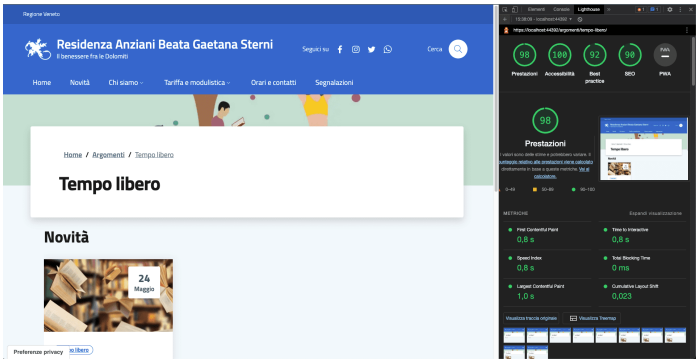
* Prestazioni $>$ 80%;
* Accessibilità al 100%;
* Best practice $>$ 90%;
* SEO $>$ 90%.

Alcuni di questi, come le prestazioni, dipendono da molti fattori a runtime, perciò la stima non è sempre ben quantificabile.
A prodotto finito, però, il sito pubblicato sul dominio di test ha garantito il raggiungimento e, anche, il superamento dei punteggi voluti (Figura 6.2a, 6.2b, 6.2c e 6.2d).
Lighthouse offre quindi, nonostante le molte variabili che possono entrare in gioco, una buona stima della qualità del prodotto ultimato, ed è stato fondamentale per il miglioramento continuo di questo.
Essendo inoltre uno strumento di *Google*, un buon punteggio in esso può essere inteso come sinonimo di un buon posizionamento nel motore di ricerca[4], che va ad osservare proprio gli stessi criteri utilizzati nell'estensione.

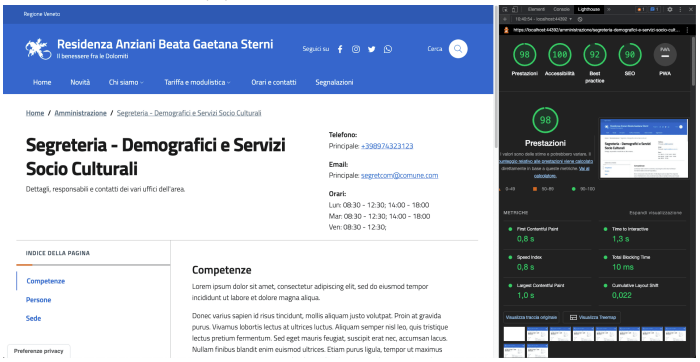
[4] *Google SEO*. URL: https://developers.google.com/search/docs/beginner/seo-starter-guide?hl=it.



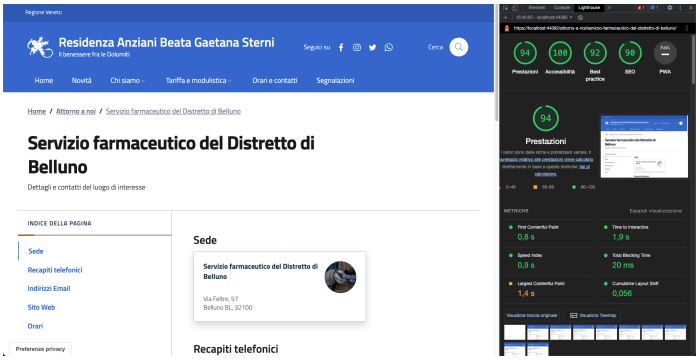
(a) Sulla HomePage



(b) Su una pagina argomento



(c) Su una pagina ufficio



(d) Su una pagina luogo

Figura 6.2: Alcuni esiti dati dallo strumento Lighthouse

6.4 Validazione

La validazione è stata effettuata solamente sul codice HTML, in quanto ogni risorsa di stile è stata presa dalla libreria *Bootstrap Italia*.
È stato quindi utilizzato lo strumento ufficiale di *Markup Validation Service*[5] del consorzio W3C, che rimanda, una volta inserito l'URL del sito, a **Nu HTML checker**[6].
A prodotto ultimato, nessun errore né warning rilevante è stato riscontrato.

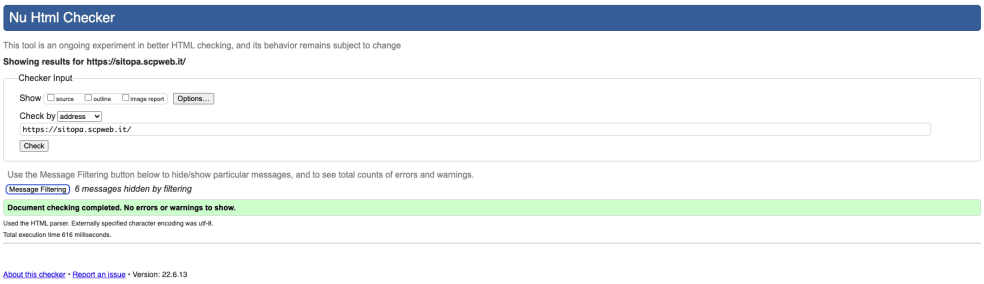
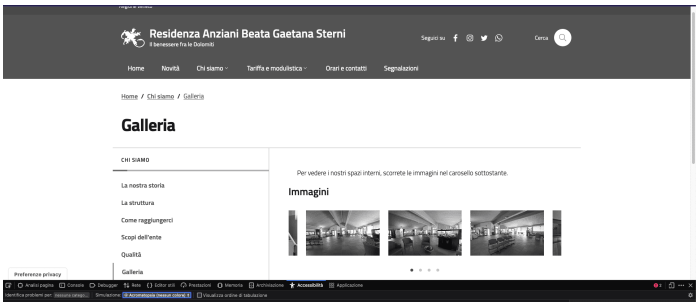


Figura 6.3: Validazione del codice finale di Markup

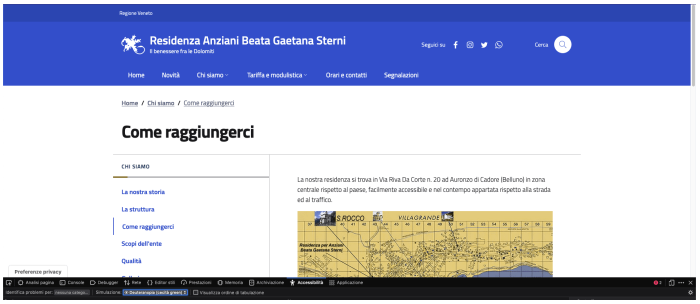
6.5 Strumenti di firefox

Sono stati effettuati alcuni test con gli strumenti per sviluppatori integrati nel browser **Firefox**.
In primo luogo si è controllato il corretto ordine della tabulazione, al fine di garantire una agevole e intuitiva navigazione del sito tramite tastiera. In alcune pagine lo strumento non riesce a dar alcun esito, pertanto in queste il test è stato eseguito in modo manuale.
Si è poi utilizzato lo strumento per simulare varie forme di cecità ai colori, per assicurare un sufficiente contrasto dei contenuti e che nessuna informazione fosse veicolata esclusivamente tramite l'uso del colore. Con questo tool si possono applicare diversi filtri, fra cui: protanopia (Figura 6.4d), deuteranopia(Figura 6.4b), tritanopia(Figura 6.4c) e acromatopsia(Figura 6.4a).
Applicando di default un contrasto elevato in tutti i contenuti, questi test manuali non hanno sollevato alcuna criticità.

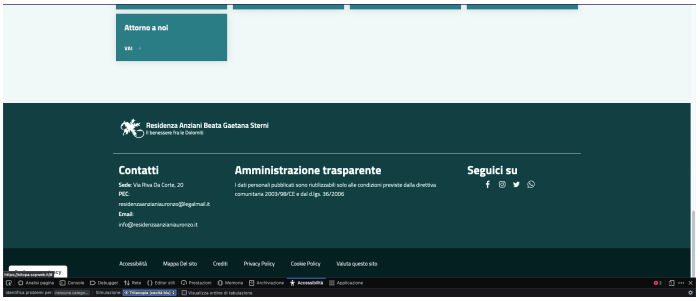
[5] *W3C Validation Service*. URL: https://validator.w3.org/.
[6] *Nu HTML Checker*. URL: https://validator.w3.org/nu/.



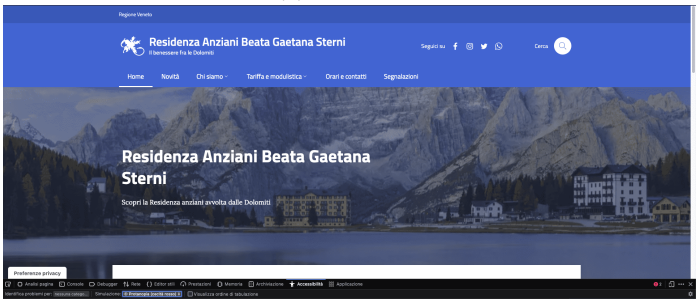
(a) Acromatopsia



(b) Deuteranopia



(c) Tritanopia



(d) Protanopia

Figura 6.4: Sito web con diversi filtri colore applicati

6.6 Screen Readers

Nel sito sono stati testati diversi screen reader:

Jaws

Testato a partire dal sistema operativo Windows 11, è uno degli screen reader più utilizzati secondo le classifiche Google. Per mezzo di questo strumento sono state testate tutte le sezioni del sito. È stato effettuato un controllo in particolare sulle abbreviazioni, sui testi alternativi, sugli orari e sui numeri di telefono che compaiono all'interno del sito. Nessuno dei controlli fatti ha rilevato criticità di comprensione, grazie agli accorgimenti apportati a priori. Il sito risulta facilmente fruibile in lettura.

VoiceOver

Il sito è stato testato con lo screen reader integrato nel sistema VoiceOver di Apple, come suggerito da un articolo del WebAIM[7]. Il medesimo tool è utilizzabile anche dagli smartphone della stessa casa produttrice, permettendo un controllo anche su dispositivi mobile. I test, a prodotto finito, non hanno evidenziato alcuna criticità.

Orca

screen reader integrato in Ubuntu, anche questo come i precedenti permette una corretta fruibilità del sito.

6.7 Test su vari dispositivi

Il sito è stato testato su molteplici categorie di dispositivo, al fine di assicurarne il corretto funzionamento in termini di aspetto e performance.

* PC (Windows, Linux, Mac) di fascia alta;
* PC (Windows, Mac) di fascia bassa;
* Smartphone di fascia alta (Android, iOS) con schermo di grandi dimensioni ($\geq$ 1920 x 1080 pixel);
* Smartphone di fascia medio/bassa (Android, iOS) con schermi di media e piccola dimensione, tra i più utilizzati secondo il sito *StatCounter*[8] (dimensione $\geq$ 360 x 800 pixel e $\leq$ 1366 x 768 pixel);
* Tablet di fascia media (Android) con dimensione schermo di 768 x 1024 pixel;
* Tablet di fascia alta (iOS) con dimensione schermo di 1668 x 2388 pixel.

Gli schermi di piccole dimensioni potrebbero portare ad alcune sovrapposizioni di testo con componenti visive. A causa delle limitazioni e la poca manutenibilità della libreria grafica utilizzata, si è tentato di risolvere dove possibile utilizzando *Media-Queries* personalizzate e troncamento di testo tramite `ellipsis`.

[7] *WebAim, uso di VoiceOver*. URL: https://webaim.org/articles/voiceover/.
[8] *Screen Resolution Stats Worldwide*. URL: https://gs.statcounter.com/screen-resolution-stats.

È stato inoltre utilizzato lo strumento *Google PageSpeed Insights*, per misurare le prestazioni del sito in un ambiente simulato.
Questo tool mostra ottime prestazioni in qualsiasi ambiente desktop, mentre evidenzia lievi difficoltà in smartphone meno performanti. Sul *device* di esempio "Moto G4" su rete 4G, ad esempio, le procedure di caricamento vengono fortemente rallentate per via della compressione del testo e la rimozione di CSS e Javascript inutilizzati. Nonostante i risultati discreti che lo strumento evidenzia su questo genere di dispositivi, nelle prove su hardware fisici non sono stati riscontrati problemi tali da inficiare l'esperienza di navigazione, che rimane scorrevole e piacevole.

6.8 Test su vari browsers

Il prodotto è stato testato sui seguenti browser, sia da mobile che da desktop:

* Chrome (v102)
* Firefox (v101)
* Opera (v86)
* Safari (v15.5)
* Edge (v102)

6.9 Esperienza con le persone

Il dominio di testing è stato veicolato ad alcune categorie di persone chiave che potessero rispecchiare le *persona* individuate. Questo ha permesso, sulla base di esperienze reali al di fuori dell'ambiente di sviluppo, di correggere eventuali criticità nell'organizzazione e nella fruibilità del sito. Alcune accortezze sono state recepite proprio a seguito di questi test, come ad esempio il *focus* automatico sulla ricerca, la disposizione di alcuni contenuti, il dimensionamento delle card e la chiarezza di navigazione. Sono state coinvolte le seguenti categorie di utenti:

* Persone di mezza età con dimestichezza nella navigazione in internet;
* Persone di mezza età senza particolare familiarità al web e i suoi standard;
* Persone anziane;
* Utenti di età varia con occhio critico e tecnico.

Si è osservato come, naturalmente, le persone anziane necessitino di un aiuto per il primo approccio al sito, per via delle difficoltà manuali e la poca familiarità con il web. Una volta introdotti però, grazie alle viste rapide che possono offrire le *card* e le sezioni ben suddivise, l'esperienza viene dichiarata piacevole e rapida nel reperimento delle informazioni più importanti.

Capitolo 7

Conclusioni

A termine delle otto settimane di tirocinio il prodotto è risultato pienamente funzionante, dotato delle funzionalità richieste e pronto all'uso. A seguito di una fase di apprendimento individuale, si è quindi sviluppato un *Template* per la creazione, la gestione e la pubblicazione di un sito web per la Pubblica Amministrazione. Tale prodotto è stato sviluppato sul CMS Umbraco: un potente Content Management System, Open-Source, leader del mercato. Basato sul framework ASP.NET core, ha permesso di costruire il *Template* da zero, basandosi sulle linee guida per i servizi digitali della Pubblica Amministrazione. Grazie al linguaggio di programmazione Razor, congiuntamente alla libreria grafica *Bootstrap Italia*, la gestione e la resa del Frontend risultano ottimali, permettendo la creazione di un sito web dinamico e con elevati standard qualitativi in termini di accessibilità, performance e funzionalità. L'uso di Umbraco facilita inoltre la gestione e la creazione del Backend, completamente personalizzabile e modellabile secondo le esigenze. Il Backoffice del prodotto distribuito, poi, risulta di facilissimo utilizzo anche per un utente amministratore a cui viene affidata la gestione finale del sito, garantendogli massima flessibilità e potenza nella gestione dei contenuti.
Il prodotto finale è stato applicato su un primo caso particolare, cioè il sito web per la residenza anziani di Auronzo di Cadore, del quale se ne riportano delle immagini dimostrative nell'elaborato. Naturalmente, un ente come una casa di riposo non necessita di tutte le sezioni e le caratteristiche che, invece, potrebbe richiedere un Comune; nonostante questo il *Template* è stato sviluppato per intero, e comprensivo di tutto ciò che necessita anche un ente più grande e complesso. È pertanto possibile utilizzare il prodotto per creare qualsiasi sito per la Pubblica Amministrazione, semplicemente distribuendo il *Template* Umbraco all'ente destinatario, il quale potrà (previa formazione) creare e gestire il proprio sito web.
L'unica funzionalità che non è stata possibile sviluppare è il login tramite identità digitale SPID, in quanto richiedeva una approvazione di carattere amministrativo non ancora disponibile durante lo sviluppo.

7.1 Consuntivo finale

La tabella di ripartizione delle ore è stata così preventivata:

Durata in ore	Descrizione dell'attività
40	**Formazione sulle tecnologie**
24	*Apprendimento Umbraco*
16	*Linee guida design per i servizi della Pubblica Amministrazione*
40	**Analisi dei requisiti**
24	*Analisi dei requisiti di un caso concreto*
8	*Presa visione delle caratteristiche dell'attuale CMS legacy*
8	*Confronto con il Product Manager*
200	**Sviluppo**
20	**Collaudo**
14	*Collaudo*
4	*Stesura documentazione finale*
1	*Incontro di presentazione della piattaforma con gli stakeholders*
1	*Live demo di tutto il lavoro di stage*
Totale ore	**300**

Data la rapida curva di apprendimento delle tecnologie, la parte di sviluppo ha richiesto 40 ore in meno, permettendo di reinvestire questo avanzo per la creazione di un manuale del prodotto. Inoltre, l'analisi dei requisiti di un caso concreto si può, a monte di tutto, integrare con lo studio delle linee guida, perchè parte di essa: ciò ha unificato di fatto le due parti richiedendo un ammontare di ore pari alla somma correttamente preventivata.

7.2 Raggiungimento degli obiettivi

A termine dello sviluppo, tutti i requisiti e gli obiettivi sono stati considerati coperti e soddisfatti.
In particolare la soluzione è stata completata e definita per intero nelle parti richieste, ottenendo ottimi risultati in termini di utilizzo, accessibilità e performance.

7.3 Conoscenze acquisite

Durante l'esperienza di tirocinio ho potuto acquisire in particolare conoscenze sul lavoro in team, immerso in un contesto reale e professionale. Le esperienze fatte fino a quel momento sono state puramente individuali o con gruppi senza esperienza, rendendo di fatto le ore di lavoro non sistematiche né tanto meno proficue rispetto agli obiettivi fissati.
Una reale esperienza in azienda mi ha insegnato a rispettare date tempistiche, requisiti e standard qualitativi ben definiti. La produttività che ne è derivata è risultata, pertanto, ben lungi da quanto avessi mai fatto prima, e questo ha avuto anche esito positivo sui fattori di efficienza e efficacia.
Le metodologie e le tecnologie utilizzate, inoltre, hanno permesso di conseguire un processo di sviluppo rapido e di qualità. In particolare, nei miei confronti, è stato utilizzato un approccio a "stimoli": ad ogni nuovo materiale da apprendere, mi veniva

dato solo un "*input*" iniziale per permettermi una formazione automa e quindi più efficace. Questa metodologia mi ha permesso di maturare proprio in un aspetto che tipicamente è più da "studente" che da "professionista", e ciò mi ha consentito di ottenere sempre una visione più profonda e ampia dei problemi e dei processi da risolvere.

7.4 Valutazione personale

L'esperienza complessivamente si può ritenere soddisfacente e altamente formativa, nonostante ci siano stati alti e bassi a livello personale.
Come detto, l'approccio molto auto formativo ha permesso di maturare sotto molti aspetti, sia personali che in termini di produttività, ma tale sistema non si è sempre rivelato efficace come dovuto, portando quindi ad un effetto opposto rispetto al voluto. Spesso, infatti, l'autonomia si è trasformata in un forte senso di spaesamento, inadatto vista l'inesperienza.
Laddove però questo non accadeva, gli aiuti e gli "input" dati dal team tecnico si sono rivelati illuminanti, e grazie anche a questo sono riuscito a sentirmi stimolato nel creare un prodotto di qualità e che potesse soddisfare i requisiti richiesti, nonché le persone dentro l'ambiente aziendale.
Provare tecnologie nuove e poter condividere idee e pensieri con dei professionisti, ha aumentato ancor di più l'interesse, la voglia di apprendere e di addentrarsi nel mondo Web e Frontend. Lavorando ad un progetto come quello in esame, inoltre, mi si è resa ancora più chiara la necessità di incentivare lo sviluppo di servizi web che siano inclusivi, funzionali e appaganti per le persone. Spesso i fattori di accessibilità vengono tralasciati, considerandoli più un obbligo di legge che una necessità. Questo naturalmente è un pregiudizio totalmente errato, e grazie a questo progetto ho consolidato ancor più l'idea che l'accessibilità e l'eleganza stilistica possano andare di pari passo ed essere complementari; con semplici e facili accorgimenti che possono rendere il web, uno strumento ad oggi fondamentale per tutti, un luogo migliore e inclusivo.

Come dichiarato in precedenza, quindi, nel complessivo posso ritenermi molto soddisfatto dell'esperienza e per quello che ho potuto apprendere, sia qui che in generale negli anni universitari. Ho potuto coltivare e approfondire una passione che finalmente prende forma in fatti concreti, e che con costanza e una maggiore maturità potrà essere, nel suo piccolo, utile a molte persone.

Glossario

.NET

framework ambiente di esecuzione runtime della piattaforma tecnologica .NET in cui vengono gestite le applicazioni destinate allo stesso .NET Framework. 41, 42

AMP

abbreviazione di *Accelerated Mobile Pages*, è un framework HTML open source sviluppato dall'AMP Open Source Project. Ottimizzato per la navigazione web mobile, è stato originariamente creato da Google allo scopo di aiutare le pagine web a caricarsi più velocemente. Tra le tecniche di ottimizzazione ci sono l'asincronia nel caricamento delle risorse esterne, l'utilizzo del sendboxing per ogni iframe, la disabilitazione di selettori CSS che risultano troppo lenti e la valutazione preventiva del layout di ciascun elemento in pagina prima di caricare effettivamente le risorse. 12

API

abbreviazione di *Application Programming Interface*, (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. 18, 41, 56

ARIA

abbreviazione di *Accessible Rich Internet Applications*,definisce uno standard per rendere più accessibile i contenuti web a tutte le categorie di utenti. 62, 104

ASP.NET core

framework open source multipiattaforma e ad alte prestazioni per la creazione di app moderne abilitate per il cloud e connesse a Internet. 41, 113

Backend

in un servizio al pubblico offerto attraverso reti telematiche o telefoniche, l'insieme delle applicazioni e dei programmi con cui l'utente non interagisce direttamente ma che sono essenziali al funzionamento del sistema. Il backend è inoltre il responsabile di elaborare i dati generati dal Frontend. 41, 64, 79, 84, 91, 95, 99, 113

Backoffice

in Umbarco, il Backoffice è la controparte del Frontend. È qui che si creano tutti i contenuti in Umbraco, gestiti nella sezione "Contenuto". È composto dall'albero delle sezioni a sinistra, la dashboard/spazio di lavoro al centro e il menu "Sezioni" in alto. 1, 21, 22, 30–37, 39, 40, 42, 45, 48, 51, 54, 59, 64, 65, 69, 71, 73, 79, 81, 83, 87, 91, 113

Bootstrap

raccolta di strumenti liberi per la creazione di siti e applicazioni per il Web. Essa contiene modelli di progettazione basati su HTML e CSS, sia per la tipografia, che per le varie componenti dell'interfaccia, come moduli, pulsanti e navigazione, così come alcune estensioni opzionali di JavaScript. 1, 19, 42, 63, 90, 104

CMS

abbreviazione di *Content Management System*, è uno strumento software, installato su un server web, il cui compito è facilitare la gestione dei contenuti di siti web, svincolando il webmaster da conoscenze tecniche specifiche di programmazione Web. 1, 4, 18, 22, 40, 41, 44, 46, 50, 87, 113

Content Type

nelle linee guida per il design dei servizi digitali per la PA si intendono tutte quelle tipologie di contenuto che possono essere utilizzare per la creazione di una pagina web. 10, 45

CSS

abbreviazione di *Cascading Style Sheets*, è un linguaggio che permette di definire lo stile delle pagine web. 19, 42, 43, 111

Data Type

in Umbraco è una tipologia di dato da associare ad una proprietà. 46, 51, 66, 69, 76, 81, 87, 88

Document Type

in Umbraco è un contenitore di dati in cui si possono aggiungere proprietà. 45, 47, 51, 57, 60, 65, 69, 72–76, 79, 82, 84, 87, 91, 93

Framework

termine della lingua inglese che può essere tradotto come struttura o quadro strutturale, in informatica e specificamente nello sviluppo software, è un'architettura logica di supporto sulla quale un software può essere progettato e realizzato, spesso facilitandone lo sviluppo da parte del programmatore. 80, 82, 89

Frontend

nel campo della progettazione software e sviluppo software il Frontend è la parte di un sistema software che gestisce l'interazione con l'utente o con sistemi esterni che producono dati di ingresso (es. interfaccia utente con un form). 40, 51, 81, 89, 91, 95, 113

HTML

abbreviazione di *HyperText Markup Language*, è un linguaggio che descrive la struttura di un documento basata sul significato semantico dei suoi elementi collocati in un albero. 11, 43, 47, 55, 81, 103

Javascript

linguaggio di programmazione multi paradigma orientato agli eventi, comunemente utilizzato nella programmazione Web lato client (esteso poi anche al lato server) per la creazione, in siti web e applicazioni web, di effetti dinamici interattivi tramite funzioni di script invocate da eventi innescati a loro volta in vari modi dall'utente sulla pagina web in uso (mouse, tastiera, caricamento della pagina ecc.). 18, 42, 111

Media-Queries

in CSS, tecnica popolare per fornire un foglio di stile su misura a diversi dispositivi. 18, 110

nested content

Data Type che permette di generare una lista di elementi analoghi. Si ottiene così il vantaggio di un'interfaccia utente riutilizzabile e familiare per creare una lista di contenuti ripetibili, contenuti in uno stesso campo dati. Questo editor restituisce un singolo elemento o una sua raccolta. 49, 50, 69, 76, 84, 86, 95

PA

abbreviazione di pubblica amministrazione, in diritto è l'insieme degli enti pubblici che concorrono all'esercizio e alle funzioni della gestione, direzione e coordinazione di uno Stato nelle materie di sua competenza. 2, 8, 16, 40, 45

Razor

sintassi di markup per incorporare codice basato su .NET in pagine Web. La Razor sintassi è costituita da Razor markup, CSharp e HTML. I file contenenti Razor in genere hanno un'estensione cshtml di file. 41, 47, 70, 84, 113

RegEx

abbreviazione di *Regular Expression* e in italiano "espressione regolare", è una sequenza di simboli che identifica un insieme di stringhe. Possono definire tutti e soli i linguaggi regolari. 69, 89

RSA

abbreviazione di "Residenza sanitaria assistenziale", sono strutture socio-sanitarie dedicate ad Anziani non autosufficienti, che necessitano di assistenza medica, infermieristica o riabilitativa, generica o specializzata. 2

RTE

abbreviazione di *Rich Text Editor*, è un editor di testo all'interno dei browser Web per la scrittura e modifica di testo RTF, che presenta all'utente un'area di modifica "ciò che vedi è ciò che ottieni". L'obiettivo è ridurre lo sforzo per gli utenti che cercano di esprimere la propria formattazione direttamente come markup HTML valido. 54, 65, 69, 73, 79, 95

screen reader

forma di tecnologia assistiva che che identifica ed interpreta il testo mostrato sullo schermo di un computer, presentandolo come output in sintesi vocale o tramite uno schermo braille. 68, 70, 84, 85, 110

template

in Umbraco è la pagina in formato `.cshtml` (CSharpHTML) che viene utilizzata per renderizzare il contenuto del Document Type. 4, 41, 42, 46, 47, 74, 76, 89

Umbraco

sistema open source di gestione per la piattaforma di pubblicazione di contenuti sul World Wide Web, scritta in C# e implementato su infrastruttura Microsoft. Il backend open source è distribuito con una licenza MIT, mentre l'interfaccia utente è commercializzata sotto la licenza umbraco. 1, 18, 22, 30–37, 40–45, 47, 48, 52, 56, 80, 81, 87, 97, 104, 113

WebP

formato di compressione per le immagini sviluppato da Google a partire dal codec video VP8. WebP è un formato di immagine moderno che fornisce una compressione lossless e lossy superiore per le immagini sul Web. Utilizzando WebP, i webmaster e gli sviluppatori web possono creare immagini più piccole e più ricche che rendono il web più veloce. 93

Bibliografia

Siti web consultati

Architettura dell'informazione Designers Italia. URL: https://docs.google.com/spreadsheets/d/1D4KbaA__xO9x_iBm08KvZASjrrFLYLKX/edit#gid=2066775910 (cit. a p. 45).

Bando "Esperienza del Cittadino nei servizi pubblici". URL: https://shorturl.at/kGMQW (cit. a p. 2).

Bootstrap 4.6. URL: https://getbootstrap.com/docs/4.6/getting-started/introduction/ (cit. a p. 104).

Bootstrap 5 Support. URL: https://getbootstrap.com/docs/5.0/getting-started/browsers-devices/ (cit. a p. 19).

Bootstrap Italia. URL: https://bootstrap-italia-next-development.vercel.app/ (cit. a p. 1).

Bootstrap Italia Next. URL: https://github.com/italia/bootstrap-italia-next (cit. a p. 19).

Browser Market Share Worldwide. URL: https://gs.statcounter.com/ (cit. a p. 19).

Certificazione Aziendale di Qualità UNI EN ISO 9001:2015. URL: https://www.scponline.it/media/1775/2021certificato-iso9001-2015.pdf (cit. a p. 2).

Estensione WAVE. URL: https://wave.webaim.org/extension/ (cit. a p. 103).

Examine. URL: https://shazwazza.github.io/Examine/ (cit. a p. 91).

GitLab. URL: https://gitlab.com (cit. a p. 44).

Google Lighthouse. URL: https://developer.chrome.com/docs/lighthouse/overview/ (cit. a p. 19).

Google PageSpeed Insights. URL: https://pagespeed.web.dev/ (cit. a p. 19).

Google SEO. URL: https://developers.google.com/search/docs/beginner/seo-starter-guide?hl=it (cit. a p. 106).

Il sito web e i servizi digitali dei Comuni italiani. URL: https://docs.italia.it/italia/designers-italia/design-comuni-docs/it/v2022.1/index.html.

ImageSharp. URL: https://sixlabors.com/products/imagesharp/ (cit. a p. 43).

Iubenda. URL: https://www.iubenda.com/it/ (cit. a p. 100).

Linee guida di design per i servizi digitali della PA. URL: https://shorturl.at/efvIS (cit. a p. 1).

Masonry. URL: https://masonry.desandro.com/ (cit. a p. 43).

Microsoft SQL Server. URL: https://www.microsoft.com/it-it/sql-server (cit. a p. 44).

MyWCAG4All. URL: https://web.math.unipd.it/accessibility-dev/ (cit. a p. 103).

Nu HTML Checker. URL: https://validator.w3.org/nu/ (cit. a p. 108).

Our Umbraco Documentation. URL: https://our.umbraco.com/documentation/ (cit. a p. 47).

SCP srl. URL: https://www.scponline.it/ (cit. a p. 2).

Screen Resolution Stats Worldwide. URL: https://gs.statcounter.com/screen-resolution-stats (cit. a p. 110).

SplideJS. URL: https://splidejs.com/ (cit. a p. 43).

Umbraco CMS. URL: https://umbraco.com (cit. a p. 1).

Visual Studio. URL: https://visualstudio.microsoft.com/it/ (cit. a p. 44).

Visual Studio Code. URL: https://code.visualstudio.com/ (cit. a p. 43).

W3C Validation Service. URL: https://validator.w3.org/ (cit. a p. 108).

WebAim, uso di VoiceOver. URL: https://webaim.org/articles/voiceover/ (cit. a p. 110).