

Università degli Studi dell'Insubria

Facoltà di Scienze MM.FF.NN.

Corso di Laurea Triennale in Informatica



Ristutturazione di un sistema di gestione dei contenuti

Relatore: Dott. Ignazio Gallo

Tesi di Laurea di
Matteo Franceschi De Marchi
Matricola 725083

Anno Accademico 2016-2017

“What a liberation to realize that the “voice in my head” is not who I am. Who am I then? The one who sees that.” - Eckhart Tolle

Indice

1	Introduzione	1
1.1	Il Kirivo Network	1
1.2	Architettura del Kirivo Network	2
1.3	I team del Kirivo Network	2
1.4	Gestione delle homepage	3
1.5	La gemma cmsdealer	3
1.6	Obbiettivo	4
1.7	Obbiettivo secondario	4
2	Fase di formazione aziendale	5
2.1	Ruby e suoi framework	5
2.2	Wordpress	5
2.3	Sviluppo agile	5
2.3.1	TDD	6
2.3.2	Pair Programming	6
3	Integrazione Page Builder	7
3.1	Page Builder by Site Origin	7
3.2	Widget	8
3.3	Inizializzazione	10
3.4	Implementazione Widget	12
4	API rendered prices	13
4.1	API	13
5	Preparazione dell'ambiente di sviluppo per Wordpress	15
5.1	Architettura per lo sviluppo	15
5.2	Procedura di deploy	16
5.3	Impostazione della macchine di sviluppo	16
5.4	Creazione dello script di deploy	17

6 Widgets di Origini	19
6.1 Origini - Slider prodotti	20
6.2 Origini - Banda tuttoschermo	21
6.3 Origini - Speciale	22
7 Widgets di Kirivo	23
7.1 Kirivo - Immagini Speciali	24
7.2 Kirivo - Immagini Newsletter	25
7.3 Kirivo - Prodotti in evidenza	26
7.4 Kirivo - Slider prodotti	27

Elenco delle figure

3.1	Il form che viene visualizzato per modificare i dati del Widget.	9
3.2	Porzione di sorgente di una pagina che utilizza il widget compilato come in 3.1.	9
6.1	Contenuto mostrato dal widget "Origni - Slider prodotti".	20
6.2	Contenuto mostrato dal widget "Origni - Banda tuttoschermo".	21
6.3	Contenuto mostrato dal widget "Origni - Speciale".	22
7.1	Contenuto mostrato dal widget "Kirivo - Immagini Speciali".	24
7.2	Contenuto mostrato dal widget "Kirivo - Immagini Newsletter".	25
7.3	Contenuto mostrato dal widget "Kirivo - Prodotti in evidenza".	26
7.4	Contenuto mostrato dal widget "Kirivo - Slider prodotti".	27

Elenco delle tabelle

1

Introduzione

Nel periodo di apprendistato in 7Pixel sono stato assegnato al team Iguana, team che si occupa della gestione principalmente front-end dei siti del *Kirivo Network*

1.1 Il Kirivo Network

Il Kirivo Network (KN) è attualmente composto da due siti www.kirivo.it e www.origini.it:

www.kirivo.it è un negozio online che vende prodotti di tutte le categorie. Il marketplace dispone di un offerta di oltre 800.000 articoli in tutte le categorie tra cui elettrodomestici, prodotti per la casa, smartphone e TV, giocattoli, moda e altri.

www.kirivo.it è il marketplace ufficiale di www.trovaprezzi.it, il principale motore di ricerca italiano per la comparazione di prezzi.

www.origini.it è una divisione verticale di Kirivo. Il sito è specializzato nella vendita di vini e offre un ampia offerta di prodotti divisi per cantine e regioni. Il sito è online da Novembre 2016.

I siti del Kirivo Network fanno utilizzo di servizi di Back-End comuni che permettono di effettuare acquisti nei due siti utilizzando un unico account ed un unico carrello.

Con buone probabilità verranno aggiunti in futuro nuovi siti verticali per alcune categorie di Kirivo.

1.2 Architettura del Kirivo Network

Per l'erogazione dei siti del Kirivo network vengono usati diversi server

- **Hybris:** una piattaforma Enterprise di e-commerce scritta in Java che offre una soluzione all-in-one per i siti di e-commerce comprendendo servizi quali la gestione del catalogo dei prodotti, degli utenti e la gestione sicura dei pagamenti. La scelta di utilizzare una piattaforma di e-commerce a pagamento è stata fatta principalmente per velocizzare i tempi di sviluppo in fase iniziale.

Hybris utilizza una database relazione Postgres e il suo cataologo viene indicizzato dal motore di ricerca SolR.

- **SolR:** un motore di ricerca scritto in Java che permette di indicizzare i prodotti presenti a catalogo per una accesso più rapido, permette inoltre di filtrare in modo efficiente i prodotti presenti a catalogo ottimizzando le ricerche per categorie o caratteristiche del prodotto.
- **Kitty:** un server di backend in Ruby, attualmente si sta lavorando nel migrare gradualmente tutte le funzionalità di Hybris in Kitty in modo da non aver più bisogno in futuro della piattaforma di e-commerce.
- **Kiruby:** un web-server Ruby che eroga le pagine web dei siti, si interfaccia con Hybris, Solr e Kitty utilizzando i loro servizi di backend.
- **Wordpress:** usato per la creazione di pagine di contenuto che vengono incluse da Kiruby.

Il server di Wordpress si trova nella LAN aziendale e si interfaccia solamente con il server Kiruby, la sua presenza è nascosta agli utentu finali.

- **Redis:** un database noSql che, salvando tutto il suo contenuto in RAM, garantisce alte prestazioni e viene usato da Kiruby come cache di contenuti, specialmente per le richieste di Kiruby a Wordpress.
- **Nginx:** un Reverse proxy usato per redirigere le chiamate fatte ai domini www.kirivo.it e www.origini.it ai server opportuni.

1.3 I team del Kirivo Network

Lo sviluppo è l manutenzione dei siti del Kirivo Network viene effettuato da più team e questi sono:

- **Team Iguana:** si occupano dello sviluppo di Kiruby.

- **Team Nimbus:** si occupano dello sviluppo di Hybris SolR e Kitty.
- **Content Manager:** si occupano della comunicazione con i venditori, della gestione dei prodotti a catalogo e della creazione di pagine di speciali, lavorano interfacciandosi con Hybris e Wordpress
- **Grafica:** lavora o direttamente su Kiruby o da al team Iguana grafiche Html che vengono poi rese dinamiche ed integrate con i vari servizi.

1.4 Gestione delle homepage

Le homepage di Origini e Kirivo sono le pagine che, nei rispettivi siti, possono cambiare contenuto più frequentemente. Inoltre scegliere quali prodotti, quali offerte e quali contenuti vanno inseriti in homepage non è compito dei programmati ma dei *content managers*, quindi si rivela importante dare la possibilità ai content di fare modifiche, come cambiare un prodotto da mettere tra quelli in evidenza in homepage, o il testo di un pannello con la cantina del mese senza dover passare dai programmati per fare modifiche.

Per dare più libertà ai content, il contenuto della homepage, ovvero tutto quello che non è header e footer, non si trova nel server Kiruby ma in pagine Wordpress che i content possono direttamente modificare accedendo alla sezione admin di Wordpress.

Il server Kiruby quando deve visualizzare la home chiede a Wordpress la pagina della home, ne estrae il contenuto e lo renderizza, nell'html tra Header e Footer nel caso delle homepage.

1.5 La gemma cmsdealer

Per visualizzare contenuti dinamici, come ad esempio un Box di 4 vini viene utilizzata una gemma Ruby chiamata *cmsdealer*.

Questa gemma viene usata dal server Kiruby. Questo mentre scansiona la pagina di Wordpress da includere se incontra un tag di nome *cmsdealer* legge l'attributo *type* e in base al valore di questo seleziona il corrispondente template, legge l'ID dei prodotti e stampa l'html del box con i valori dei prodotti selezionati

Esempio: se nella pagina Wordpress Kiruby trova

```
<dynamic type="OriginilistingBox" ids="3422,2345,2872,2209" />
```

allora verrà cercato il template di *originilistingbox.html.erb* e verrà popolato coi valori dei prodotti con gli id specificati nell'attributo *ids*.

1.6 Obbiettivo

L'obbiettivo del progetto è quello di rendere l'edit delle pagine da parte dei content molto più semplice e flessibile, facendo in modo che i contenuti delle homepage possano essere editati visualmente e non andando a mettere mano direttamente sul codice html.

Per farlo i content dovranno interagire con un interfaccia grafica web che permette la modifica delle informazioni necessarie e di poter spostare e copiare componenti della home con un click o con un drag and drop.

1.7 Obbiettivo secondario

Obbiettivo secondario del progetto e di rendere il sito più manutenibile.

La criticità del sistema è che contiene codice duplicato nella gemma CmsDealer, in Kiruby e nei widget di Wordpress.

Per risolvere il problema sono state create le *RenderdPricesAPI* che restituiscono frammenti di html renderizzato per varie componenti della home.

In questo modo il codice del template resta solamente in Kiruby e esponendo le API questo viene usato sia dai Widget di Wordpress sia dalla gemma CmsDealer

2

Fase di formazione aziendale

Durante i primi due mesi di tirocinio, la maggior parte del mio tempo è stato speso in formazione, il mio studio può essere diviso in tre argomenti principali:

- Tecnologie
- Sviluppo agile

Lo studio dei primi due è stato prevalentemente pratico, mentre per lo sviluppo agile lo studio è stato in gran parte teorico ma anche pratico

2.1 Ruby e suoi framework

All'inizio il linguaggio

Poi kata agili (con framework di testing)

Poi redis and co

2.2 Wordpress

2.3 Sviluppo agile

In 7Pixel si sviluppa utilizzando metodologie agili, il software viene sviluppato iterativamente, nuove feature e aggiornamenti vengono pubblicati quotidianamente. Nel periodo di studio

2.3.1 TDD

Una tecnica rigorosamente usata in azienda è il TDD Test Driven Development ovvero, prima di aggiungere una qualsiasi nuova funzionalità si scrive un test che passa solo se quella funzionalità fosse implementata.

Dopodichè si cerca nel modo più veloce e semplice possibile di fare passare il test

Una volta passato il test si fa del refactoring per rendere il codice più leggibile e soprattutto manutenibile

2.3.2 Pair Programming

Tutti i lavori effettuati su Kiruby sono sempre stati fatti in Pair Programming con un membro del team

Il Pair Programming si rivela molto efficace, perchè si riescono ad evitare molti errori "di distrazione" che possono costare caro in termini di tempo e soprattutto si ha molto spesso la possibilità di confrontarsi con punti di vista diversi che possono portare ad un'analisi più approfondita del problema e a soluzioni migliori.

Dal mio punto di vista di apprendista il Pair Programming ha portato inoltre il vantaggio di poter lavorare con gente più esperta e quindi, durante il lavoro, di imparare tecniche metodologie e "best practices" per lo sviluppo.

3

Integrazione Page Builder

Per passare ad una gestione più semplice delle pagine da parte dei Content mi è stato chiesto di cercare qualche sistema che permetesse una suddivisione della pagina in componenti facilmente editabili e riutilizzabili.

Le esigenze principali erano:

- modificare il contenuto delle componenti da interfaccia grafica e non editando il codice HTML.
- poter creare semplicemente copie delle componenti già create
- poter spostare le varie componenti con *drag and drop* e avere feedback visivo immediato della modifica della pagina

Per questo mi è stato consigliato di fare una ricerca tra le varie soluzioni disponibili nella ampia libreria di plugin di Wordpress.

Dopo aver analizzato i vari plugin disponibili è stata individuata una soluzione open source chiamata *Page Builder by Site Origin*[1] che soddisfava le esigenze.

3.1 Page Builder by Site Origin

Il plugin *Page Builder* permette la suddivisione della pagina in colonne all'interno delle quali si possono inserire delle componenti.

Queste componenti possono essere duplicate, modificate aprendo il form della componente e spostate con drag and drop.

Il vantaggio di questo plugin rispetto ad altri plugin che permettono la suddivisione delle pagine è che all'interno delle colonne oltre a poter inserire delle componenti standard fornite dal plugin si possono anche inserire i Widget di Wordpress.

3.2 Widget

I *Widget* sono delle componenti riutilizzabili che possono essere aggiunte a qualsiasi tipo di pagina Wordpress.

Classici esempi di Widget sono le icone per i link ai social network, o un anteprima di un post creato, tuttavia è possibile creare Widget personalizzati.

Per creare un widget bisogna implementare una sottoclassa di `WP_Widget`[2] e sovrascrivere il metodo `widget` dove viene restituito il contenuto HTML che la pagina deve restituire quando quel widget viene incluso e, se si vuole del contenuto dinamico, bisogna sovrascrivere `form` dove viene creato il form HTML che verrà visualizzato dai content per modificare le parti dinamiche della componenti utilizzata da `innin`.

Le componenti compilate nel form vengono poi utilizzate dal metodo `widget`.

```

1  <?php
2  class simpleWidget extends WP_Widget {
3      function __construct() {
4          parent::__construct( false , 'Kirivo - Simple Widget' );
5      }
6
7      function widget( $args , $instance ) { ?>
8          <div>
9              <h1>Titolo del widget : <?php echo $instance['title']?></h1>
10             </div>
11             <?php
12         }
13
14     function form( $instance ) {?>
15         <p>
16             <label for=<?php echo $this->get_field_id('title');?>"'
17                 'Titolo:');</label>
18             <input class="widefat" id=<?php
19                 echo $this->get_field_id('title');?>
20                 name=<?php echo $this->get_field_name('title');?>
21                 type="text" value=<?php echo $instance['title'];?>" />
22         </p>
23         <?php
24     }
25 }
```

Una semplice implementazione di un widget che stampa un titolo dinamicamente

The screenshot shows a configuration interface for a 'Simple widget'. On the left, there's a title input field with the value 'My title'. On the right, there are four tabs: 'Widget Styles' (selected), 'Attributes', 'Layout', and 'Design'. At the bottom, there are 'Delete' and 'Duplicate' buttons, and a 'Done' button.

Figura 3.1: Il form che viene visualizzato per modificare i dati del Widget.

```
<div>
  <h1>Titolo del widget : My title</h1>
</div>
```

Figura 3.2: Porzione di sorgente di una pagina che utilizza il widget compilato come in 3.1.

3.3 Inizializzazione

Per inizializzare i Widget bisogna aggiungere una action[3], ovvero un meccanismo offerto da Wordpress per aggiungere dei comportamenti prestabiliti, chiamata *init_widget* dove viene aggiunto ogni Widget creato.

Per far in modo di non dover aggiungere manualmente un Widget al file di inizializzazione ogni volta che se ne crea uno nuovo, è stato implementato un script di inizializzazione che va all'interno delle cartelle *widget_orgini* e *widget_kirivo* e aggiunge tutti i Widget che trova all'interno di queste cartelle, per individuare un widget all'interno della cartella viene fatto pattern-matching per i file che terminano con *Widget.php*.

```

1 <?php
2 function getKirivoWidgets() {
3     $res = array();
4     foreach(glob( dirname(__FILE__)."/widget-kirivo/*Widget.php")
5             as $filename) {
6         array_push($res,$filename);
7     };
8     return $res;
9 }
10 function getOrginiWidgets() {
11     $res = array();
12     foreach(glob( dirname(__FILE__)."/widget-orgini/*Widget.php")
13             as $filename) {
14         array_push($res,$filename);
15     };
16     return $res;
17 }
18 function my_register_widgets() {
19     foreach(getKirivoWidgets() as $filename) {
20         require_once $filename;
21         register_widget(str_replace('.php','','',str_replace(dirname(__FILE__),
22                     '/widget-kirivo/','','',$filename)));
23     }
24     foreach(getOrginiWidgets() as $filename) {
25         require_once $filename;
26         register_widget(str_replace('.php','','',str_replace(dirname(__FILE__),
27                     '/widget-orgini/','','',$filename)));
28     };
29 }
30 add_action( 'widgets_init', 'my_register_widgets' );

```

Il file initialize-widget.php

3.4 Implementazione Widget

Per implementare i Widget necessari per le Homepage sono state isolate le componenti HTML necessarie e per ognuna di queste è stato creato un Widget poi , per ognuna di queste è stato create il form per le componenti che potevano essere modificate dai Content.

Ci sono due categorie principali di widget

- Widget che non contengono prodotti.
- Widget che contengono prodotti.

Per implementare i Widget che non contengono prodotti è stato messo all'interno del metodo *widget* direttamente l'HTML di quella componente con eventuale contenuto dinamico stampato con php.

Per i widget che contengono prodotti invece all'interno di *widget* non è stato inserito il codice HTML con il template di ogni prodotto ma è stato messo il tag *dynamic* i cui attributi sono popolati dinamicamente in base al contenuto del form. Una volta che il server Kiruby prende una pagina Wordpress e trova il dynamic tag allora provvede a sostituire il tag con le 'tile' di tutti i prodotti specificati.

4

API rendered prices

Nel periodo di apprendistato[?] in 7Pixel sono stato assegnato al team Iguana, team che si occupa della gestione principalmente front-end dei siti del *Kirivo Network*

4.1 API

Il Kirivo Network (KN) è attualmente composto da due siti www.kirivo.it e www.origini.it:

5

Preparazione dell'ambiente di sviluppo per Wordpress

Wordpress, essendo un Content Management System open source, è altamente personalizzabile e dispone di un innumerevole quantità di plugin, gratuiti e a pagamento, per ogni tipo di funzionalità.

Al mio arrivo veniva usata un'installazione base di Wordpress con la sola aggiunta di un plugin "Microsoft Azure for Wordpress" che fa in modo che tutte le immagini che vengono caricate dal pannello di admin di Wordpress vengono caricate e servite da un server di Microsoft Azure.

Questo serve principalmente, come accennato precedentemente, ad oscurare il dominio del server di Wordpress, infatti se in una pagina del KN fossero linkate le immagini con l'URL di Wordpress questo potrebbe comportare dei problemi di sicurezza.

Per iniziare i miei lavori su Wordpress era necessario impostare un ambiente di sviluppo e dei modi per automatizzare la distribuzione delle modifiche.

Inoltre era necessario aggiungere un sistema di versionamento che fino a quel momento per Wordpress, a differenza degli altri progetti, non veniva utilizzato.

5.1 Architettura per lo sviluppo

In 7Pixel per lo sviluppo di tutte le applicazioni, come tipico delle aziende che fanno sviluppo agile, viene usata la seguente architettura basata su tre macchine:

- **Macchina di produzione:** è la macchina da cui vengono servite l'applicazione per l'utente finale.
- **Macchina di LAB:** è un ambiente identico a quello di produzione. Viene utilizzato per testare le nuove funzionalità prima di venire deploiate in produzione.
- **Macchina di sviluppo:** è la macchina dove lavorano gli sviluppatori, non vengono usati dati reali per i prodotti, ma solo un numero ridotto di prodotti fake utili ai fini di testing.

5.2 Procedura di deploy

Per la distribuzione delle modifiche si usa il seguente procedura

- **Sviluppo in locale:** viene editato il codice per aggiungere una nuova funzionalità usando TDD, una volta visto in locale che la funzionalità è stata implementata correttamente si passa alla fase successiva
- **Test in lab:** le nuove modifiche vengono deploiate in LAB, qui, sfruttando un ambiente simile a quello di produzione, vengono fatti ulteriori controlli, se si riscontra qualche problema si ritorna alla fase precedente e si corregge altrimenti si passa alla fase successiva
- **Deploy in produzione:** una volta effettuati i controlli in LAB le modifiche vengono pubblicate sulle macchine di produzione e saranno disponibili agli utenti finali, nei minuti successivi si tiene sotto controllo **New Relic**, un'applicazione di monitoraggio degli errori, per vedere se le modifiche pubblicate fanno generare degli errori inaspettati. In caso di errori si fa "rollback" alla versione precedente altrimenti la nuova funzionalità viene considerata pubblicata con successo

Prima dei deploy, sia in LAB che in produzione, vengono fatti girare tutti i test, unitari e di integrazione, e il codice viene pubblicato solo se tutti questi sono "verdi".

5.3 Impostazione della macchine di sviluppo

Prima del mio arrivo il Team Iguana non si occupava dello sviluppo di Wordpress, non era quindi presente l'applicazione nelle macchine di sviluppo locale.

è stato mio compito quindi, prima di iniziare a sviluppare, di installare su tutte le macchine di sviluppo un'istanza di Wordpress, servita dal server Nginx, lo stesso server già presente nelle macchine locali per la reindirizzazione delle chiamate a Kiruby.

è stato poi creato un repository di git per il versionamento di Wordpress. Alla radice della cartella "wordpress" troviamo alcuni file di configurazione e le cartelle "wp-content", "wp-admin", ... di queste solo la cartella "wp-content" viene versionata perché

qui troviamo tutti i file rilevanti per la programmazione delle varie funzionalità, mentre nelle altre cartelle troviamo file di configurazione e altre funzionalità standard di "wordpress" che non vengono modificati dagli sviluppatori.

5.4 Creazione dello script di deploy

Lo script di deploy, essendo tutte la macchine di sviluppo e produzione macchine Linux, è stato scritto in un file eseguibile "kirCMS-autodeploy.sh" utilizzando il linguaggio Bash ed utilizza una procedura simile allo script di deploy utilizzato da Kiruby.

Prima di eseguire il deploy è necessario fare commit sul master della repository di Wordpress in modo da avere i sorgenti git aggiornati.

Una volta committate le modifiche da pubblicare bisogna eseguire lo script "kirCMS-autodeploy.sh" passandogli come argomento "lab" o "pro" a seconda della macchina su cui si vuole deploiare.

Lo script per prima cosa chiede le credenziali di autenticazione, apre un tunnel verso la macchina dove si vuole deploiare e si collega a questo in remoto.

Una volta collegato in remoto viene scaricata da "Gitlab" l'ultima versione committata della repository dentro la cartella /tmp della macchina e vengono eseguiti tutti i test di PHPUnit eseguendo lo script "test.sh".

Se i test falliscono viene stampato a console un messaggio con lo stack dell'errore e la procedura termina, lasciando invariata la versione precedentemente pubblicata.

Se tutti i test passano la cartella scaricata in /tmp viene spostata dentro wordpress col nome wp-content e il server inizierà a utilizzare le ultime modifiche, la precedente cartella wp-content viene rinominata come wp-content-bkp, la precedente cartella wp-content-bkp viene eliminata.

La cartella wp-content-bkp serve a tenere ancora disponibile la versione precedente semplificando la procedura per un eventuale rollback. Se vengono individuati degli errori con il nuovo deploy, per ritornare alla situazione precedente è sufficiente eliminare la cartella wp-content e rinominare wp-content-bkp in wp-content.

Comando di rollback "/media/www/wordpress. rm wp-content ... mv wp-content-bkp wp-content"

6

Widgets di Origini

LE SELEZIONI

I CONSIGLI DEL NOSTRO SOMMELIER


Cuveè Spumante Extra Brut

Montello e Colli Asolani DOC
Giusti Wine
📍 Veneto

Bottiglia 0,75 lt **16,47 €**


Syrah

Terre Siciliane IGT
Tenuta San Giaime
📍 Sicilia

Bottiglia 0,75 lt **15,00 €**


Traminer Aromatico

Friuli Grave DOC
Fossa Mala
📍 Friuli-Venezia Giulia

Bottiglia 0,75 lt **9,80 €**


Cortona

Cortona DOC
Palazzo Vecchio
📍 Toscana

Bottiglia 0,75 lt **11,00 €**

[Scopri Altro](#)

Figura 6.1: Contenuto mostrato dal widget ”Origini - Slider prodotti”.

6.1 Origini - Slider prodotti

Il Widget ”Origini - Slider prodotti” visualizza un box contenente 4 vini, con titolo, sottotitolo e link per ”Scopri altro”.

Il form permette di modificare i seguenti campi (riferirsi a Figure 6.1):

- Sottotitolo: il testo ”Le selezioni”
- Titolo: il testo ”I consigli del sommelier”
- Ids prodotti: la lista degli ID dei prodotti da visualizzare separati da virgola
- Link: l’URL a cui punta il tasto scopri



Figura 6.2: Contenuto mostrato dal widget ”Origini - Banda tuttoschermo”.

6.2 Origini - Banda tuttoschermo

Il Widget ”Origini - Banda tuttoschermo” visualizza quella porzione di HTML usata attualmente per lo ”Speciale regione” (vedi Figure 6.2).

Il form permette di modificare i seguenti campi (riferirsi a Figure 6.2):

- Sottotitolo: il testo ”Speciale regione”
- Titolo: il testo ”Lombardia e cultura del buon vino”
- Descrizione: il testo della descrizione ”La varietà dei territori...”
- Link: l’URL a cui punta il tasto scopri
- Image-Link: l’URL dell’immagine di sfondo



Figura 6.3: Contenuto mostrato dal widget "Origini - Speciale".

6.3 Origini - Speciale

Il Widget "Origini - Speciale" visualizza quella porzione di HTML usata attualmente per lo "Speciale del mese" (vedi Figure 6.3).

Il form permette di modificare i seguenti campi (riferirsi a Figure 6.3):

- Sottotitolo: il testo "La cantina del mese"
- Titolo: il testo "Federico Ferrero"
- Descrizione: il testo della descrizione "L'azienda agricola Ferrero nasce..."
- Link: l'URL a cui punta il tasto scopri
- Image-Link: l'URL dell'immagine da visualizzare

7

Widgets di Kirivo



Figura 7.1: Contenuto mostrato dal widget "Kirivo - Immagini Speciali".

7.1 Kirivo - Immagini Speciali

Il Widget "Kirivo - Immagini Speciali" permette di modificare il contenuto delle prime due immagini della Homepage, ovvero le immagini solitamente dedicate allo speciale del mese e alle offerte

I campi che si possono modificare sono (riferirsi a Figure ??):

- Link speciale: l'URL dove si viene indirizzati quando si schiaccia sull'immagine dello speciale
- Url immagine speciale desktop: l'url dell'immagine da mettere nello speciale per desktop
- Url immagine speciale mobile: l'url dell'immagine da mettere nello speciale per mobile
- Link offerte: l'URL dove si viene indirizzati quando si schiaccia sull'immagine delle offerte
- Url offerte speciale desktop: l'url dell'immagine da mettere nelle offerte per desktop
- Url offerte speciale mobile: l'url dell'immagine da mettere nelle offerte per mobile



Figura 7.2: Contenuto mostrato dal widget "Kirivo - Immagini Newsletter".

7.2 Kirivo - Immagini Newsletter

Il Widget "Kirivo - Immagini Newsletter" permette di modificare il contenuto delle prime due immagini della Homepage, ovvero le immagini solitamente dedicate allo speciale del mese e alle offerte

I campi che si possono modificare sono (riferirsi a Figure 7.2):

- Url immagine newsletter desktop: l'url dell'immagine da mettere nella sezione "iscriviti alla newsletter"
- Url immagine newsletter mobile: l'url dell'immagine da mettere nella sezione "iscriviti alla newsletter"
- Link offerta: l'URL dove si viene indirizzati quando si schiaccia sull'immagine dell'offerta
- Url offerta desktop: l'url dell'immagine da mettere nelle offerte per desktop
- Url offerta mobile: l'url dell'immagine da mettere nelle offerte per mobile

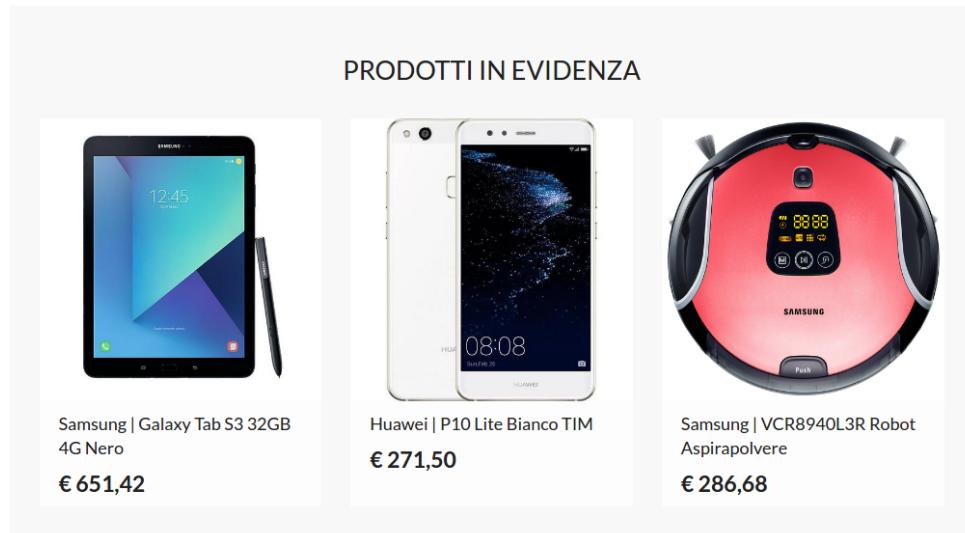


Figura 7.3: Contenuto mostrato dal widget "Kirivo - Prodotti in evidenza".

7.3 Kirivo - Prodotti in evidenza

Il Widget "Kirivo - Prodotti in evidenza" visualizza quella porzione di HTML usata per visualizzare i prodotti in evidenza (vedi Figure 7.3).

I campi che si possono modificare sono (riferirsi a Figure 7.3):

- Titolo: il testo "PRODOTTI IN EVIDENZA".
- Ids: l'ID dei prodotti da visualizzare separati da virgola. I prodotti devono essere almeno tre, se sono più di tre verranno usati i primi 3 ID validi.



Figura 7.4: Contenuto mostrato dal widget "Kirivo - Slider prodotti".

7.4 Kirivo - Slider prodotti

Il Widget "Kirivo - Slider prodotti" visualizza quella porzione di HTML usata per visualizzare uno slider di un insieme di prodotti (vedi Figure 7.4).

I campi che si possono modificare sono (riferirsi a Figure 7.4):

- Titolo: il testo "PRODOTTI PIÙ POPOLARI".
- Numero max: il numero massimo di prodotti da visualizzare. Se lasciato vuoto non viene imposto alcun limite.
- Ids: l'ID dei prodotti da visualizzare separati da virgola. I prodotti devono essere almeno quanti specificati in numero max o quattro se non viene specificato.

Colophon

La tesi è stata scritta utilizzando il linguaggio LaTeX.

Le immagini sono state create appositamente usando screenshots delle applicazioni ed eventualmente editate con GIMP.

Bibliografia

- [1] “Siteorigin page builder,” <https://siteorigin.com/page-builder/>.
- [2] “Widgets api,” https://codex.wordpress.org/Widgets_API.
- [3] “Plugin api/action reference,” https://codex.wordpress.org/Plugin_API/Action_Reference.