**Indice**

[**Introduzione** 1](#_Toc13689785)

[**Descrizione problematiche riscontrate** 2](#_Toc13689786)

[**Vantaggi della migrazione** 2](#_Toc13689787)

[**Analisi di impatto** 3](#_Toc13689788)

# **Introduzione**

L’applicazione web per la gestione delle incombenze legate al GDPR in via di sviluppo è stata scritta utilizzando principalmente quattro linguaggi di programmazione: HTML e CSS per gestire il lato front-end, PHP per gestire il lato back-end, quindi la comunicazione con la base di dati, e JavaScript per il passaggio dei dettagli relativi agli eventi alle librerie di FullCalendar (utilizzate per la visualizzazione del calendario all’interno delle varie schermate). Al momento della scelta iniziale su quale fosse il miglior modo per aiutare il cliente a risolvere una parte del problema che ci ha presentato si è deciso di sviluppare un’applicazione web con i quattro linguaggi sopra citati in quanto erano quelli più adatti per arrivare in tempi brevi a sviluppare un primo prototipo che potesse soddisfare le esigenze del cliente. In particolare, è stato scelto PHP perché i membri del team avevano già una certa familiarità con questo linguaggio, inoltre PHP permette di ottenere risultati concreti con uno sforzo richiesto molto basso.

# **Descrizione problematiche riscontrate**

Il primo dei problemi riscontrato dal team di sviluppo riguarda l’architettura del sistema: il primo prototipo costruito dal team utilizzando PHP (versione 0.1 del codice consegnata al cliente) non aveva un’architettura ben definita. Il passaggio dei parametri fra il front-end ed il back-end avveniva tramite frammenti di codice PHP inserito all’interno dei file HTML che contenevano le varie schermate dell’applicazione. Il codice che gestiva le varie comunicazioni con il database (inserimento e modifica dei dettagli di un evento) veniva tenuto in file in cui oltre alle query (inserimento e modifica) erano presenti anche le varie validazioni (data inizio minore della data di fine, utente fra i partecipanti, eccetera) fatte con una serie di if a cascata. In questo modo il codice risultava caotico e non rispettando il Single Responsability Principle che stabilisce che ogni modulo, classe o funzione ha una responsabilità su una singola parte di una funzionalità del sistema software.

Il secondo problema riguarda la gestione delle interazioni fra PHP e FullCalendar. FullCalendar è una libreria scritta in JavaScript che mette a disposizione la possibilità di utilizzare un calendario Drag and Drop all’interno di un’applicazione. Questa libreria si aspetta di ricevere un array (JavaScript) di eventi dove ogni evento ha (almeno) tre campi: titolo, data inizio e data fine. Dal momento che la comunicazione con il database avviene utilizzando PHP, per passare gli eventi al file JavaScript che permette di fare il rendering del calendario abbiamo inserito all’interno di una pagina HTML, un frammento di codice PHP che andava a prendere i dettagli degli eventi e li salvava in un array che poi veniva letto da uno script JavaScript presente all’interno della stessa pagina HTML e poi convertito in un formato leggibile da FullCalendar. Anche in questo caso abbiamo diverse operazioni che dal punto di vista logico andrebbero separate, inoltre questo metodo di passaggio dei parametri è sconsigliato e andrebbe gestito in modo migliore, ad esempio utilizzando richieste Ajax.

Il terzo problema è parzialmente collegato al secondo e riguarda sempre il passaggio di parametri fra PHP e FullCalendar. Dal momento che il documento dei requisiti specifica che deve essere possibile modificare i dettagli relativi agli eventi e che il cliente ha chiesto la possibilità di accedere al form di modifica di un evento tramite click dal calendario le possibili soluzioni sono due: utilizzare un ascoltatore jQuery che al click della casella relativa all’evento sul calendario effettui una richiesta Ajax passando l’id dell’evento al form per la modifica oppure definire per ogni evento un url a cui andare quando si clicca sulla casella dell’evento sul calendario. In questo secondo caso bisognerebbe usare come url il path del file PHP contenente il form di modifica e a questo url andrebbe aggiunto in coda un parametro del tipo “&id” che permette al form di avere l’id dell’evento in modo da poter effettuare una query tramite PHP per precompilare il form con i suoi dettagli. Anche queste operazioni potrebbero essere gestite in modo migliore, soprattutto in un’ottica in cui in fasi avanzate dello sviluppo del prodotto vengano introdotti nuovi requisiti che potrebbero complicare la struttura (già complessa per un numero molto ridotto di funzionalità) del sistema.

Oltre a questi problemi i membri del team di sviluppo, che nel frattempo hanno utilizzato il framework Rails in altri progetti, hanno notato che la gestione dell’applicazione potrebbe essere semplificata di molto sfruttando le funzionalità messe a disposizione dal framework.

# **Vantaggi dell’utilizzo del framework Rails**

Quando ci si trova a dover sviluppare un’applicazione, una scelta può essere quella di appoggiarsi ad un framework: un framework è un’infrastruttura che fornisce un certo livello di supporto durante lo sviluppo di un sistema software; in questo modo i programmatori possono astrarre da alcuni dettagli (ad esempio come costruire un’architettura per l’applicazione) e sfruttare le funzionalità messe a disposizione dal framework. Nel caso specifico di Rails, l’architettura su cui è costruito il framework è un’architettura MVC (Model, View, Controller) dove il Model è la componente che si occupa della gestione delle comunicazioni con la base di dati, il Controller si occupa del passaggio dei dati fra il Model e le viste, mentre le View sono appunto le viste ovvero le schermate dell’applicazione. Queste tre componenti permettono di ottenere un’architettura stabile e mantenibile in cui ogni componente ha le sue responsabilità; il framework funziona bene utilizzando questa architettura quindi è sufficiente che il programmatore rispetti le convenzioni che sono alla base dell’utilizzo del framework per ottenere un’applicazione con una struttura solida e mantenibile. Ad esempio, una di queste convenzioni, per quello che abbiamo appena detto sull’architettura MVC, è quella di evitare di inserire troppi frammenti di codice all’interno delle viste in quanto queste dovrebbero solo utilizzare i dati ricevuti dal Controller o dall’utente e non dovrebbero contenere al loro interno una parte della logica dell’applicazione.

1. Validazioni a livello di modelli;
2. Gestione routes;
3. Gestione semplificata full calendar;
4. Generazione automatica parte documentaz;
5. Test driven;
6. Gemme per integrare librerie ausiliarie (login gratis);
7. Buona conoscenza del framework da parte dei membri del team => poco tempo di sviluppo e risultati buoni;
8. Generaz codice automatico.

Uno degli svantaggi che potrebbe derivare dall’utilizzo di un framework è lo sforzo richiesto per imparare ad utilizzare le sue funzionalità (almeno quelle principali) e questo è stato il motivo per cui il team ha scelto inizialmente di utilizzare degli strumenti che già conosceva.

# **Analisi di impatto**

1. Possibili problemi: comunicaz con fullcalendar (gemma), utilizzo libreria per gestione delle notifiche (cdn), login (gemma)