

UNIVERSITÀ DEGLI STUDI DI PERUGIA
Dipartimento di Matematica e Informatica

CORSO DI LAUREA TRIENNALE IN INFORMATICA



Tesi di Laurea

**Gestione automatizzata dei dati sensibili di una
pubblica amministrazione**

Laureando:

Andrea Pompili

Relatore:

Prof. Osvaldo Gervasi

Dott. Damiano Perri

Dott. Marco Simonetti

Anno Accademico 2019–2020

*Alla mia Famiglia
e ai miei Amici*

Indice

Introduzione	v
1 Il GDPR e il problema dei dati sensibili	2
1.1 Protezione dei dati: come si è sviluppata in Europa	4
1.1.1 Diritto alla privacy	4
1.1.2 Protezione dei dati	5
1.1.3 Le prime norme di protezione dei dati	6
1.1.4 La legislazione sulla protezione dei dati negli anni '90 e nei primi anni 2000	7
1.1.5 Il Regolamento generale sulla protezione dei dati del- l'Unione Europea	9
1.2 GDPR	9
1.3 Garante della privacy	11
1.4 DPO	12
2 Le principali tecnologie utilizzate	14
2.1 Oracle VM VirtualBox	14
2.2 Laravel	16
2.2.1 Il pattern MVC	17

2.2.2	I Controllers	17
2.2.3	Route	18
2.2.4	Le View	19
2.2.5	Sicurezza e protezione	21
2.3	phpMyAdmin	22
2.4	PHP	23
2.4.1	Variabili	24
2.4.2	Funzioni	25
2.5	HTML	26
2.5.1	Sintassi	27
3	Ingegnerizzazione del software e lo schema architetturale	
	proposto	30
3.1	Registrazione	32
3.2	Login	33
3.3	Pagina Home	34
3.4	Modifica tabelle	35
3.4.1	Inserisci record	36
3.4.2	Elimina record	37
3.4.3	Modifica record	38
3.5	Schede di trattamento	40
3.5.1	Visualizzazione schede	42
4	Descrizione dell'applicazione realizzata	45
4.1	Backend: ambiente e funzioni	45
4.2	Il database MySQL	46

4.2.1	istruzioni SQL	47
4.2.2	Struttura del database	48
4.2.3	gestione del database tramite Laravel	57
Conclusioni		61
Bibliografia		65

Elenco delle figure

2.1	Logo di Oracle VM VirtualBox	15
2.2	Logo di Laravel	16
2.3	MVC	17
2.4	Logo di phpMyAdmin	23
3.1	Registrazione	32
3.2	Login	33
3.3	Diversi menu utente	34
3.4	Sezione di edit.blade.php	35
3.5	Form di inserimento	36
3.6	delete_t.blade	37
3.7	Form di aggiornamento	38
3.8	Prima parte del flusso	40
3.9	Bottoni sotto ai form	41
3.10	Visualizzazione delle tabelle dei form	43
3.11	Esempio di presentazione della scheda in modalità ag- giornamento	44
4.1	Schema E/R del database	49

Introduzione

Nella presente tesi, verranno mostrate diverse fasi per la realizzazione del progetto GDPRegistry, l'applicativo presentato è un sito web gestito dall'Università degli Studi di Perugia con lo scopo di compilare una scheda del trattamento dei dati personali in modo di rendere più agevole il controllo del rispetto delle normative europee (GDPR, regolamento generale per la protezione dei dati) sul mantenimento e sulla gestione dei dati delle persone coinvolte nelle attività universitarie.

Questo progetto è stato sviluppato sotto richiesta della Dottoressa Flavia Cristiano, Data Protection Officer (DPO) di Ateneo, che ci ha affidato delle linee guida per lo svolgimento del lavoro e inoltre ci ha informati sugli aspetti normativi e giuridici necessari per poter affrontare il programma. E' stato perciò esplorato tutto il contorno legislativo sulla protezione dei dati, la definizione di privacy e l'evoluzione storica dei soggetti e dei termini ad essi collegati, la creazione e la formazione dei vari strumenti internazionali e come si è arrivati alla creazione di un regolamento europeo fino alla nascita del GDPR (Regolamento 2016/679) e dei vari soggetti ad esso collegati, tra questi il Garante della protezione dei dati e il soggetto RPD, ovvero il responsabile della protezione dei dati andando a guardare le sue responsabilità e funzioni anche in relazione allo strumento da noi realizzato.

La realizzazione del progetto di Tesi è stata affidata a me e al collega di corso Carlo Genovesi. La realizzazione si è articolata in una suddivisione dei compiti, che ci ha permesso di ottenere e condividere le conoscenze su tutti gli strumenti utilizzati e le problematiche affrontate nel progetto.

L'obiettivo dell'applicazione è quello di fornire uno strumento che possa essere utilizzato dagli agenti designati dall'Università per la gestione del trattamento dei dati personali nei vari servizi da questa offerti. Questo si concretizza mediante la compilazione di schede tramite le quali sarà possibile ottenere un indice di rischio e la valutazione di come questi servizi sono svolti. L'analisi dei dati elaborati consente ai soggetto interessati di avere una visione chiara del come avviene la gestione dei dati personali e l'associato livello di rischio.

Capitolo 1

Il GDPR e il problema dei dati sensibili

L'applicativo descritto in questo elaborato è stato sviluppato sulla base delle normative europee sulla salvaguardia dei dati personali. I dati personali sono le informazioni che identificano o rendono identificabile una persona fisica e che possono fornire informazioni sulle sue caratteristiche, le sue abitudini, il suo stile di vita, le sue relazioni personali, la sua situazione economica, il suo stato di salute, ecc..

Particolarmente importanti sono:

- i dati che permettono l'identificazione diretta - come i dati anagrafici (ad esempio: nome e cognome), le immagini, ecc. - e i dati che permettono l'identificazione indiretta, come un numero di identificazione (ad esempio, il codice fiscale, l'indirizzo IP, il numero di targa);
- i dati rientranti in particolari categorie: si tratta dei dati "sensibili", cioè quelli che rivelano l'origine razziale od etnica, le convinzio-

ni religiose, filosofiche, le opinioni politiche, l'appartenenza sindacale, relativi alla salute o alla vita sessuale.

- i dati relativi a condanne penali e reati: si tratta dei dati "giudiziarri", cioè quelli che possono rivelare l'esistenza di determinati provvedimenti giudiziari soggetti ad iscrizione nel casellario giudiziale (ad esempio, i provvedimenti penali di condanna definitivi, la liberazione condizionale, il divieto od obbligo di soggiorno, le misure alternative alla detenzione) o la qualità di imputato o di indagato. [1]

Con l'evoluzione delle nuove tecnologie, altri dati personali hanno assunto un ruolo significativo, come quelli relativi alle comunicazioni elettroniche (via Internet o telefono) e quelli che consentono la geo-localizzazione, fornendo informazioni sui luoghi frequentati e sugli spostamenti.

Quando si parla di dati sensibili entrano in gioco tre parti:

- Interessato è la persona fisica alla quale si riferiscono i dati personali. Quindi, se un trattamento riguarda, ad esempio, l'indirizzo, il codice fiscale, ecc. di Mario Rossi, questa persona è l'"interessato";
- Titolare è la persona fisica, l'autorità pubblica, l'impresa, l'ente pubblico o privato, l'associazione, ecc., che adotta le decisioni sugli scopi e sulle modalità del trattamento;
- Responsabile è la persona fisica o giuridica alla quale il titolare richiede di eseguire per suo conto specifici e definiti compiti di gestione e controllo per suo conto del trattamento dei dati. [1]

1.1 Protezione dei dati: come si è sviluppata in Europa

1.1.1 Diritto alla privacy

Ambiti in cui le informazioni personali sono state subordinate a specifiche norme di riservatezza sono sempre esistiti. Gli esempi classici sono costituiti dal giuramento di Ippocrate del 4° secolo a.C. per la professione medica, e dal “sigillo della confessione” per la Chiesa Cattolica romana. In tempi più recenti, in particolare a partire dal XIX secolo, a banchieri, avvocati, altri ministri del culto, lavoratori postali e delle telecomunicazioni e molti altri professionisti è stato chiesto di trattare le informazioni ricevute in ambito lavorativo come riservate, privilegiate, o addirittura sacrosante. Tuttavia, benché per certe categorie ci fosse un dovere di riservatezza, non esisteva un diritto corrispondente per pazienti, clienti o cittadini, di verificare l'accuratezza e la rilevanza dei dati che li riguardavano. [5]

Il diritto alla “privacy” o al “rispetto della vita privata” è stato sancito solo nei Trattati internazionali sui diritti umani del secondo dopoguerra, come il Patto internazionale dell'ONU sui diritti civili e politici (ICCPR, Art. 17) e la Convenzione europea sui diritti umani (ECHR, Art. 8), volta a tutelare soprattutto dalle illecite interferenze dello Stato nella vita privata del singolo, pensiamo alle intercettazioni delle comunicazioni da parte di organi di Stato o la criminalizzazione di atti della sfera sessuale privata.

In sintesi: le legislazioni e le normative sulla riservatezza, il segreto professionale e la segretezza e le garanzie derivate dai diritti dell'uomo in materia di privacy e vita privata non hanno tutelato, e non tutelano, adeguata-

mente i singoli dalla raccolta e dall'utilizzo abusivo dei loro dati personali. Per questo motivo, in tempi più recenti, è stato riconosciuto quale diritto separato e distinto il diritto alla “protezione dei dati personali” (“protezione dei dati”).

1.1.2 Protezione dei dati

Il bisogno di proteggere i diritti umani e le libertà nei regimi democratici in relazione al trattamento automatizzato dei dati personali è emerso solo più tardi quando, negli anni '60, i computer hanno cominciato ad essere usati a fini gestionali, sia nel settore pubblico che in quello privato. Per gli alti costi e lo spazio che occupavano all'epoca, solo i paesi più sviluppati investirono sullo sviluppo dei computer, e spesso solo a beneficio dei maggiori organismi pubblici e delle aziende. I primi utilizzi dei computer riguardavano il pagamento dei salari e dei fornitori, i registri dei pazienti negli ospedali, i censimenti pubblici e le statistiche.

Alla luce di questi sviluppi, alla fine degli anni '60/inizi anni '70, si cominciò a parlarne all'OCSE e al Consiglio d'Europa. All'inizio i dibattiti si svolgevano tra esperti, vincolati da precisi obblighi etici, e fra politici, preoccupati dei rischi di abuso, uso improprio o sicurezza dei dati trattati in modo automatizzato.

Il termine “protezione dei dati” (in tedesco: *Datenschutz*) venne coniato per il titolo della primissima legge in materia, risalente al 1970, la Legge sulla protezione dei dati (*Datenschutzgesetz*) del Land tedesco dell'Assia, redatta dal “padre della protezione dei dati”, il Professore Spiros Simitis. Però il titolo utilizzava un termine improprio, dal momento che la Legge non

proteggeva i dati, ma il diritto degli individui i cui dati venivano trattati. [2]

1.1.3 Le prime norme di protezione dei dati

La primissima legge al mondo sulla protezione dei dati è stata la *Datenschutzgesetz* del Land tedesco dell'Assia, adottata nel settembre 1970. La legge introdusse anche il primo Organismo indipendente di vigilanza della protezione dei dati. In quello stesso decennio seguirono, in Europa, l'adozioni di leggi di protezione dei dati nazionali in Svezia (1973), la prima Legge federale tedesca sulla protezione dei dati (alla fine del 1977), la Legge francese Informatica e Libertà del 6 gennaio 1978, leggi in Austria, Danimarca e Norvegia (tutte del 1978) e Lussemburgo (1979).

Le leggi varate negli anni '70 in Europa si sono articolate intorno ad un "nucleo" di principi e di diritti, i quali trovarono un riflesso nei primi strumenti europei (non-vincolanti) in materia ad opera del Consiglio d'Europa. I principi "fondamentali" furono poi riconosciuti in strumenti internazionali globali, ma non ancora vincolanti, quali:

- Le Linee-guida sulla protezione della vita privata e sui flussi transfrontalieri di dati personali dell'OCSE del 1980;
- I Principi guida per la regolamentazione dei file di dati personali computerizzati dell'ONU del 1989, adottati dall'Assemblea Generale dell'ONU (UNGA).

Il primo strumento internazionale vincolante nel campo della protezione dei dati è stata la Convenzione sulla protezione delle persone rispetto

al trattamento automatizzato dei dati a carattere personale del Consiglio d'Europa del 1981, meglio nota come Convenzione sulla protezione dei dati (CPD) o “Convenzione N°108” dalla numerazione della Serie dei Trattati Europei. In qualità di strumento internazionale vincolante, la Convenzione del 1981 (a differenza dei precedenti strumenti non vincolanti) doveva includere, e utilmente lo fece, definizioni giuridiche più precise dei concetti fondamentali della legislazione sulla protezione dei dati personali: “dati personali”, “titolare del trattamento” e “trattamento”. La Convenzione ha aggiunto anche un articolo specifico sul trattamento di “categorie speciali di dati”, cioè “i dati personali che rivelano le origini razziali, le opinioni politiche, le convinzioni religiose o altre convinzioni, nonché i dati a carattere personale relativi alla salute o alla vita sessuale” e “i dati a carattere personale relativi a condanne penali”, i cosiddetti dati sensibili. [5]

1.1.4 La legislazione sulla protezione dei dati negli anni '90 e nei primi anni 2000

Per un certo periodo, la Comunità Europea ritenne che la Convenzione sulla protezione dei dati personali del Consiglio d'Europa del 1981 garantisse, in questo campo, una tutela sufficiente. Alla fine del decennio, però, emerse chiaramente che la Convenzione non avrebbe portato ad una maggiore o più armonizzata protezione dei dati personali nella Comunità: al settembre del 1990, solo sette Stati membri della Comunità economica europea l'avevano ratificata, e la legislazione di questi Stati divergeva in maniera considerevole su alcuni aspetti fondamentali.

L'anno seguente, nel settembre del 1990, la Commissione Europea pre-

sentò una serie di proposte ambiziose e complesse finalizzate alla protezione dei dati personali in tutto il territorio della Comunità Europea. Il pacchetto includeva proposte per due Direttive nell'ambito del Primo pilastro:

- una Direttiva generale della Comunità Europea “relativa alla tutela delle persone fisiche con riguardo al trattamento dei dati personali”, che dopo un iter legislativo piuttosto lungo ha dato origine alla Direttiva sulla protezione dei dati dell'Unione Europea;
- un'ulteriore Direttiva di carattere sussidiario della Comunità Europea “sulla protezione dei dati personali nel contesto delle reti pubbliche di telecomunicazione digitale”, divenuta poi la Direttiva sulla protezione dei dati nel settore delle telecomunicazioni

Più avanti la Commissione decise di intraprendere una revisione generale del quadro di regolamentazione delle comunicazioni elettroniche alla luce delle nuove tecnologie e prassi aziendali in via di definizione. Uno dei risultati della revisione fu la proposta, nel 2000, di sostituzione della Direttiva sulla protezione dei dati nelle telecomunicazioni con una nuova Direttiva riguardante la protezione dei dati nel settore delle comunicazioni elettroniche. Questo portò all'adozione, nel luglio 2002, della Direttiva sulla tutela della vita privata nel settore delle comunicazioni elettroniche, la Direttiva 2002/58/CE, abitualmente definita “Direttiva e-Privacy”. [3]

1.1.5 Il Regolamento generale sulla protezione dei dati dell'Unione Europea

Alla fine del primo decennio del XXI sec. è emerso chiaramente che gli strumenti fondamentali di protezione dei dati del XX sec., di cui abbiamo parlato sopra, non erano più sufficienti: erano stati concepiti e redatti prima del massiccio accesso a Internet, a dispositivi mobili, “Big Data”, processi decisionali algoritmici e “Intelligenza Artificiale”. Sia a livello del Consiglio d'Europa che dell'Unione Europea, vennero allora elaborati nuovi o “aggiornati” strumenti di protezione.

La Commissione Europea ha proposto l'adozione di un Regolamento generale sulla protezione dei dati (RGPD) nel 2012, per rispondere alle sfide lanciate dalle nuove tecnologie e dai servizi annessi. Fu subito chiaro che sarebbe stato necessario un livello forte ed elevato di protezione dei dati come condizione indispensabile per creare un clima di fiducia nell'ambiente online che in sé costituisce un fattore chiave dello sviluppo economico;

1.2 GDPR

Il nuovo Regolamento europeo sul trattamento e la gestione dei dati dei cittadini europei che è diventato applicabile il 25 maggio 2018 prendendo il posto del Codice della Privacy (D. Lgs. n.196/2003). Il Regolamento, insieme alla Direttive UE 2016/680 e 681, fa parte del pacchetto di riforma europea sulla protezione dei dati, nello specifico, per il trattamento dei dati personali svolti nelle attività di indagine per il perseguimento dei reati.

Nello specifico cosa cambia?

Il GDPR amplia e rafforza significativamente le disposizioni e le norme principali; aggiunge espressamente i dati genetici e biometrici al catalogo dei dati “sensibili”; è finalizzato ad una maggiore armonizzazione della legislazione sulla protezione dei dati negli Stati membri dell’Unione Europea; stabilisce diritti più forti (e in alcuni casi nuovi) per i soggetti interessati; favorisce una cooperazione transfrontaliera molto più stretta fra le DPA degli Stati membri e dovrebbe comportare una migliore e più coerente applicazione e osservanza delle norme.

Più nel dettaglio, il GDPR introduce: la definizione di dato personale costituente oggetto del Regolamento (art. 4 GDPR): è dato personale qualsiasi informazione concernente una persona fisica identificata o identificabile, include quindi i dati genetici, dati relativi alla salute, relazioni sociali, economici e finanziari, giudiziari; *privacy by design* e *privacy by default* (art. 25 GDPR): i processi aziendali che comportano il trattamento dei dati personali devono essere progettati fin dall’inizio con l’obiettivo di riduzione dei rischi per gli interessati; la prestazione del consenso: la prestazione del consenso deve essere espressa dall’interessato mediante un atto libero, specifico, informato e inequivocabile. Inoltre, rispetto al vecchio Codice Privacy, l’informativa deve essere molto più dettagliata, deve indicarne le finalità, quali sono i soggetti che hanno accesso ai dati e deve essere fornita a tutti gli interessati al trattamento.

Nel GDPR, come nella precedente normativa, il controllo spetta al Garante della Privacy, che nell’effettuare gli opportuni controlli si avvale anche della cooperazione del nucleo ispettivo della guardia di finanza. Inoltre istituisce una figura del tutto nuova e obbligatoria, il DPO, se sussistono

le seguenti condizioni: se il trattamento dei dati personali è effettuato da un'autorità pubblica o un organismo pubblico; quando le attività principali dell'organizzazione consistono in trattamenti che, richiedono il monitoraggio regolare e sistematico degli interessati su larga scala; quando le attività principali dell'organizzazione consistono nel trattamento dei dati sensibili ovvero giudiziari su larga scala. Il Responsabile designato è chiamato a mediare tra gli interessati, il titolare del trattamento e il Garante per la Privacy.

All'interno del Regolamento è specificato che nel caso di una violazione dei dati personali oggetto di trattamento (perdita, accesso non autorizzato), va:

- Notificata al garante entro 72 ore;
- Notificata a tutti gli interessati i cui dati siano stati violati (nei casi più gravi).

1.3 Garante della privacy

Il Garante per la protezione dei dati personali è un organo collegiale, composto da quattro membri eletti dal Parlamento, i quali rimangono in carica per un mandato di sette anni non rinnovabile. L'attuale Collegio è stato eletto dal Parlamento il 14 luglio 2020 e si è insediato il 29 luglio 2020, ed è composto da:

- Pasquale Stanzone (Presidente)
- Ginevra Cerrina Feroni (Vice Presidente)

- Agostino Ghiglia (Componente)
- Guido Scorza (Componente)

Il Garante per la protezione dei dati personali si occupa, tra l'altro di controllare che i trattamenti di dati personali siano conformi al Regolamento nonché a leggi e regolamenti nazionali e prescrivere, ove necessario, ai titolari o ai responsabili dei trattamenti le misure da adottare per svolgere correttamente il trattamento nel rispetto dei diritti e delle libertà fondamentali degli individui. L'organo dovrà collaborare con le altre autorità di controllo e prestare assistenza reciproca al fine di garantire l'applicazione e l'attuazione coerente del Regolamento. Nel caso di trattamenti che violano le disposizioni del Regolamento, dovrà: rivolgere ammonimenti al titolare e del trattamento o al responsabile del trattamento e ingiungere di conformare i trattamenti alle disposizioni del Regolamento; imporre una limitazione provvisoria o definitiva del trattamento, incluso il divieto di trattamento; ordinare la rettifica, la cancellazione di dati personali o la limitazione del trattamento; e in fine tenere registri interni delle violazioni più rilevanti e imporre sanzioni pecuniarie ove previsto dal Regolamento e dalla normativa nazionale. [4]

1.4 DPO

Con l'istituzione dei DPO, che anche se non ufficializzati erano già presenti nella prassi e nelle normative degli stati membri, si va a creare una nuova figura che coordinando gli sforzi con il Garante ha aiutato alla creazione di una Rete di Responsabili della protezione dei dati.

Questi soggetti a livello giuridico sono delle figure completamente nuove che aiutano a fortificare e a rendere efficace il principio di responsabilizzazione, infatti la nomina di un DPO e il vasto numero di compiti e doveri che lo attendono saranno utili all'ottenimento di un maggiore rispetto delle disposizioni del GDPR, e parallelamente sarà anche un capace alleato delle DPA che possono vedere in questo soggetto un contatto diretto, e già molte di queste lo hanno reso una figura prioritaria.

Tra le sue referenze, oltre a quella di essere un elemento altamente professionale anche e soprattutto in materia di protezione dei dati e giuridica, il Responsabile si dimostra anche capace di risolvere le problematiche che gli sono state assegnate dagli agenti esterni e superiori. Questo insieme di aspetti pratici solitamente, viene accompagnato da una serie di linee guida di tipo personale, infatti la scelta di un DPO viene presa anche in conformità ad una serie di caratteristiche personali che lo rendono la persona perfetta per svolgere quel tipo di ruolo, dato che comunque è fondamentale un atteggiamento che sia corretto anche nei momenti di scontro tra il titolare del trattamento e il Rappresentante. [5]

Capitolo 2

Le principali tecnologie utilizzate

Per lo sviluppo dell'applicativo sono state utilizzate le seguenti tecnologie:

- Oracle VM VirtualBox per l'esecuzione della macchina virtuale;
- Framework Laravel;
- phpMyAdmin per amministrare il database MySQL;
- linguaggio PHP per la programmazione lato server;
- linguaggio HTML per gestire le viste;
- CSS e framework Bootstrap per gli stili delle viste.

2.1 Oracle VM VirtualBox

Oracle VM VirtualBox è un'applicazione di virtualizzazione multiplatforma, serve per estendere la capacità del computer esistente in modo che si possano eseguire più sistemi operativi allo stesso tempo. [10]



Figura 2.1: Logo di Oracle VM VirtualBox

Nel nostro caso è stata usata per la configurazione del programma. Al suo interno è stato configurato il sistema operativo Ubuntu 18.04, dove sono stati installati una serie di software, tra questi:

- PHP7, linguaggio di scripting open source utilizzato per lo sviluppo web;
- Apache2, server web che si occupa di rispondere a richieste di risorse HTTP;
- Composer, gestore delle dipendenze per PHP, semplifica il lavoro dello sviluppatore rendendo facile l'utilizzo di librerie di terze parti;
- Docker, piattaforma che permette di rilasciare l'applicativo in un contenitore software;
- RaiDrive, un'applicazione che permette di mappare i servizi cloud su Windows. Nel nostro caso è stato utilizzato per accedere alla memoria della macchina virtuale;



Figura 2.2: Logo di Laravel

- PuTTY, client SSH, Telnet e SFTP per Windows. Utilizzato per l'accesso remoto ai computer server su una rete utilizzando il protocollo SSH;
- PuTTYgen, strumento per la creazione di chiavi SSH per PuTTY;

2.2 Laravel

La scrittura del progetto è stata svolta tramite Laravel.

Laravel è un framework PHP open source creato nel 2011 da Taylor Otwell e giunto alla versione 8.5.15. Lo scopo del framework è facilitare la programmazione web, oltre a fornire un comodo sistema di interazione con i database.

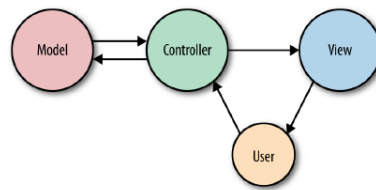


Figura 2.3: MVC

2.2.1 Il pattern MVC

Laravel usa il pattern architetturale MVC - Model View Controller, in cui ci sono tre parti del framework che funzionano insieme: model, view e controller. I model rappresentano le tabelle all'interno del database, questi serviranno da appoggio ai controller per effettuare le operazioni di modifica e inserimento. I controller, che all'interno del progetto sono indicizzati nella cartella `Http/Controllers`, prendono le richieste HTTP dal browser e i dati corretti dal database. I controller si occupano anche della validazione degli input dell'user e dell'eventuale invio di una risposta a schermo. Infine, le view serviranno per rappresentare le viste che mostrano l'output dei dati, prese tramite i controller, in un template in formato HTML. [6]

2.2.2 I Controllers

Nel pattern MVC, i controller sono essenzialmente classi che organizzano la logica di una o più route insieme in un unico punto. I controller tendono a raggruppare insieme percorsi simili, un controller potrebbe gestire tutte le azioni che possono essere eseguite su una particolare risorsa. Il compito principale resta quello di acquisire l'intento di una richiesta HTTP e trasmetterla al resto dell'applicazione. [7]

Per la creazione di un controller bisogna eseguire un comando Artisan, eseguendo dal terminale la seguente sintassi: `php artisan create:controller nomeController`, questo creerà un nuovo file chiamato `nomeController.php`, che avrà questa forma.

Listing 2.1: Controller generato di default

```
1 <?php
2
3 namespace App\Http\Controllers;
4 use Illuminate\Http\Request;
5
6 class nomeController extends Controller{
7     //
8 }
```

2.2.3 Route

Il framework supporta il meccanismo del routing, un termine con il quale ci si riferisce alla gestione delle URL e il mapping tra esse e le funzionalità offerte dall'applicazione. In un'applicazione Laravel, si possono definire delle Route (o rotte) nella directory `routes/web.php`. Queste serviranno per fare il collegamento tra le View e i controller.

La definizione di una rotta permette di creare una relazione tra una funzione presente nel controller ed un oggetto della vista, ad esempio un tasto per sottoscrivere un form. Tramite queste chiamate, infatti è possibile definire le varie funzioni che dovranno essere connesse agli input dell'utente

(GET, POST, DELETE, PUT) ed è possibile anche passare dei parametri da un metodo all'altro. [8]

Una delle funzionalità più comode, usata molto spesso all'interno del progetto in questione, è la possibilità di definire rotte parametriche. La forma di una chiamata di una funzione in un controller con il passaggio di parametri è di questo tipo: `Route::get('viewForm/{serial?}', 'App\Http\Controllers\fillFormController@viewForm')->name('viewForm');`, dove possiamo notare diversi elementi:

- La definizione di una nuova rotta con la parola chiave `Route::`;
- il metodo richiamato, in questo caso il metodo `GET`, al cui interno c'è l'url dell'indirizzo che verrà usato e la variabile passata dalla funzione precedente (`$serial`), questa in particolare è una variabile opzionale;
- il nome del metodo richiamato (`viewForm`), il quale si trova nel percorso specificato prima dell'operatore `@`;
- il nome della rotta, con cui verrà richiamata all'interno dell'applicazione.

2.2.4 Le View

Le View (o viste) rappresentano il componente del framework che si occupa di definire la struttura delle pagine HTML e dei dati che verranno serviti agli utenti. In Laravel, ci sono due formati di visualizzazione che si possono usare: PHP semplice o i modelli Blade.

Blade è il motore di creazione di modelli, semplice ma potente, incluso in Laravel. A differenza di alcuni motori di template PHP, Blade non ti

impedisce di usare il semplice codice PHP. In effetti, tutti i modelli Blade vengono compilati in semplice codice PHP e memorizzati nella cache fino a quando non vengono modificati, il che significa che Blade aggiunge essenzialmente zero overhead alla tua applicazione. I file modello Blade utilizzano l' `.blade.php` estensione del file e sono generalmente archiviati nella `resources/views` directory.

Le viste blade possono essere restituite dalle rotte o dal controller utilizzando l'helper globale `view`, definendole in questo modo:

Listing 2.2: Rotta per richiamare una vista

```
1 Route::get('/', function () {  
2     return view('greeting', [ 'name' => 'Finn' ] );  
3 });
```

È possibile visualizzare i dati passati alle viste Blade racchiudendo la variabile tra parentesi graffe. Ad esempio, dato il percorso precedente, si può visualizzare all'interno della pagina il contenuto della variabile `name` in questo modo: `{{ $name }}`.

Blade fornisce anche comode scorciatoie per le strutture di controllo PHP comuni, come istruzioni condizionali e cicli. Queste scorciatoie forniscono un modo molto chiaro e conciso di lavorare con le strutture di controllo PHP, le quali vengono richiamate usando il carattere `@`:

Listing 2.3: struttura di controllo condizionale

```
1 @foreach ( $users as $user )  
2     @if ( $user->type == 1 )  
3         //
```

```
4      @endif
5
6      <li>{{ $user->name }}</li>
7
8      @if ($user->number == 5)
9          @break
10     @endif
11 @endforeach
```

In questo esempio si possono notare l'uso delle strutture di controllo condizionale, che all'interno del Blade sono state utilizzate per stampare o meno a schermo le sezioni giuste da mostrare all'utente. Inoltre, altre shortcut sono usate per altri scopi, come `@error()`, `@enderror` le quali vengono utilizzate nella compilazione dei form che collegate alla funzione `validate()` restituiscono il messaggio di errore se viene inserito un messaggio errato. Un'altra shortcut molto utilizzata nel progetto è `@extend()`, metodo che permette di estendere un layout di una blade superiore, nel nostro caso usata per estendere su tutti i blade il menù utente e la navbar in alto. Infine altre due shortcut sono state utilizzate all'interno dei blade e sono `@user` e `@guest`, elementi che verificano se la persona che sta visitando il sito lo sta facendo come utente o come ospite.

2.2.5 Sicurezza e protezione

Un altro concetto importante svolto dal framework è quello della protezione e sicurezza, svolto grazie ai token CSFR e lo strumento Query-builder.

Laravel genera automaticamente un "token" CSRF per ogni sessione utente attiva gestita dall'applicazione. Questo token viene utilizzato per verificare che l'utente autenticato sia la persona che effettivamente effettua le richieste all'applicazione. Poiché questo token viene archiviato nella sessione dell'utente e cambia ogni volta che la sessione viene rigenerata, un'applicazione dannosa non è in grado di accedervi. Ogni volta che si definisce un modulo HTML "POST", "PUT", "PATCH" o "DELETE" nella propria applicazione, è necessario includere un token CSRF (@csrf) nascosto nel modulo in modo che il middleware di protezione CSRF possa convalidare la richiesta.

Il query-builder di Laravel fornisce un'interfaccia comoda e fluida per creare ed eseguire query di database. Può essere utilizzato per eseguire la maggior parte delle operazioni di database nella tua applicazione. Il generatore di query Laravel utilizza l'associazione dei parametri PDO per proteggere l'applicazione dagli attacchi di SQL injection. Non è necessario pulire o controllare le stringhe passate al generatore di query. Un esempio di query-builder può essere espresso dalla seguente funzione: `$user = DB::table('users')->where('name', 'John')->first();` [9]

2.3 phpMyAdmin

Per gestire il database è stata usata phpMyAdmin, un'applicazione open source che consente di amministrare un database MySQL. Grazie alla sua interfaccia grafica permette di creare un database da zero, creare, modificare e cancellare intere tabelle o singoli record. Inoltre ha delle funzionalità per



Figura 2.4: Logo di phpMyAdmin

l'inserimento dei dati, per l'esportazione e l'importazione di dati, e molte altre funzioni.

2.4 PHP

PHP è l'acronimo di "Hypertext Preprocessor", è un linguaggio open source lato server utilizzato per creare Pagine Web dinamiche. E' un linguaggio di Scripting interpretato, il cui interprete è un Software Libero distribuito mediante una licenza PHP.

PHP si differenzia poi dai linguaggi di scripting client side, come per esempio JavaScript, per via del fatto che il sorgente viene eseguito nel server, generando markup HTML da inviare al client; quest'ultimo riceverà i risultati dell'esecuzione di un'applicazione ma il codice eseguito non sarà mai visibile.

Sostanzialmente è un linguaggio di alto livello, dato il suo alto numero di funzioni API presenti nel nucleo base (oltre 3000), ma prende alcune

caratteristiche da C che gli permettono di agire anche a basso livello.

Permette di interfacciarsi con diversi DBMS, tra i quali MySQL e PostgreSQL e solitamente il suo utilizzo è affiancato a quello di un linguaggio lato Client (come ad esempio Javascript), in quanto i due linguaggi sono complementari tra di loro. Infatti, essendo PHP lato server, non è in grado di interagire con gli eventi del Client, quindi ha bisogno di un linguaggio di scripting di questo tipo, che gli permetta di capire quando dovranno essere effettuate le diverse operazioni.

Un documento di questo tipo, si può creare con un qualsiasi editore di testo, al suo interno il codice php deve essere racchiuso tra il tag di apertura `<?php` e quello di chiusura `?>`, in quanto l'interprete PHP prenderà in considerazione solo quella parte. Successivamente il file dovrà essere salvato in un formato `.php`. [12]

2.4.1 Variabili

Le variabili sono elementi che possono memorizzare dei valori di diverso tipo, per dichiararle si indicano con `$` seguito dal nome della variabile (non può iniziare con un numero). In questo linguaggio, il tipo delle variabili non viene indicato, in quanto il tutto è svolto automaticamente in base al valore che inizializzerà la variabile. Infatti PHP non è un linguaggio fortemente tipizzato, ovvero una variabile può assumere, nel corso del programma, diversi valori di diversi tipo.

Esistono diverse tipi di variabili, ognuna con un proprio scope, ovvero la parte di codice dove la stessa variabile può essere utilizzata una volta dichiarata, tra queste troviamo:

- Variabili locali : sono dichiarate all'interno di una funzione e non possono essere utilizzate al di fuori di essa.
- Variabili globali : sono dichiarate fuori dalle funzioni e possono essere utilizzate in qualsiasi parte del programma, tranne che nelle funzioni. L'unico modo per utilizzarle nelle funzioni, è dichiararle tramite la parole chiave 'global'.
- Variabili statiche : dichiarate all'interno delle funzioni con la parola chiave 'static' e possono essere utilizzate anche dopo che termina il blocco della funzione in cui sono dichiarate, quindi il suo contenuto non viene cancellato.
- Variabili Superglobali : sono delle variabili che sono sempre disponibili in tutte le parti del programma. Sono racchiuse in degli array predefiniti, quindi per accedervi si utilizzano le parentesi quadre indicando il nome della variabile da utilizzare al suo interno. [11]

2.4.2 Funzioni

Come in ogni altro linguaggio di programmazione, le funzioni sono create per eseguire delle operazioni ogni volta che vengono richiamate.

Nel caso del PHP, per dichiararle, si utilizza la parola chiave "**function**", seguito dal nome della funzione, tra parentesi tonde la lista dei parametri e successivamente si indicherà il codice da eseguire tra parentesi graffe. Inoltre possono avere un valore di ritorno indicato dall'espressione affiancata alla parola chiave '**return**'.

In questo linguaggio il tipo non è indicato, nè per la funzione, nè per i parametri della funzione.

Inoltre PHP mette a disposizione dell'utente, diverse funzioni predefinite, nel nostro caso abbiamo utilizzato:

- `var_dump($var)`: serve a stampare a schermo il contenuto della variabile, utilizzata a livello di debug;
- `isset($variable)`: controlla se una variabile è impostata, il che significa che deve essere dichiarata e non è NULL, restituisce true se la variabile esiste e non è NULL, altrimenti restituisce false;
- `count($array)`: restituisce il numero di elementi in un array;
- `max($array)`: restituisce il valore più alto in un array o il valore più alto di diversi valori specificati.
- `array_sum($array)`: restituisce la somma di tutti i valori nell'array.

[11]

2.5 HTML

HTML è l'acronimo di HyperText Markup Language. E' un linguaggio di Markup, solitamente utilizzato per la formattazione dei documenti ipertestuali nell'Internet, ovvero un testo che può contenere al suo interno diversi tipi di contenuti (testo, immagini, video ecc...).

Con il passare degli anni, il suo utilizzo principale è diventato quello per dividere la struttura logica di una pagina web dalla sua rappresentazione, gestita appunto tramite CSS.

L'ultima versione disponibile è la 5 ed è stata rilasciata nel 2014 dal W3C. Con l'ultima versione, è stata garantita una maggiore compatibilità tra i diversi browser, indipendentemente dalla piattaforma utilizzata.

In generale, come già detto nella definizione, questo non è un linguaggio di programmazione, infatti non può definire variabili, funzioni, strutture dati etc., ma è un linguaggio di Markup e supporta l'inserimento di script e oggetti esterni.

Solitamente il suo contenuto è disponibile all'interno di una pagina Web, infatti per renderlo visibile, il browser deve scaricare tale documento, lo interpreta e lo rende visibile all'utente. [14]

2.5.1 Sintassi

In un documento HTML, un testo può essere immesso tra 2 etichette dette Tag, entrambe sono identificate con un nome messo tra parentesi angolari, con una piccola differenza che le contraddistingue, la prima etichetta è detta tag di apertura e si indica con `<nometag>`, mentre la seconda etichetta è detta di chiusura e si indica con `</nometag>`. Ogni etichetta di apertura deve avere un'etichetta di chiusura, che ne indichi la fine, tranne in alcuni casi particolari.

In un documento HTML, esistono dei tag che devono essere sempre presenti e servono per definire la struttura della pagina:

- `<!DOCTYPE html>`: è il primo tag che compare nel documento ed è l'unico scritto in maiuscolo e non ha un tag di chiusura. Tramite questo elemento, il file è identificato come HTML;

- **<html>**: E' inserito subito dopo il tag visto in precedenza, racchiude al suo interno tutto il codice HTML e necessita di un tag di chiusura.
- **<head>**: deve avere un tag di chiusura ed al suo interno può racchiudere diversi tag, che forniscono informazioni sul documento e impartiscono direttive al browser, come ad esempio Titolo, fogli di stile e script. Tutti questi elementi non saranno visibili agli occhi dell'utente (ad eccezione del titolo).
- **<body>**: necessita di un tag di chiusura, al suo interno sono immessi tutti gli elementi che saranno visualizzati nella pagina, quindi tutti i tag al suo interno, compreso body, possono avere uno o più attributi che indicano come devono essere visualizzati.

Nel nostro caso, oltre a quelli obbligatori, sono stati utilizzati i seguenti tag:

- **<select>** e **<option>** servono per far apparire i menu a tendina e visualizzare le opzioni, usato sia nella fase di registrazione dell'utente per scegliere la lingua e il dipartimento e nella compilazione delle schede di valutazione per rispondere alle varie domande.
- **<h1>** usato per definire le intestazioni o per evidenziare una parte di testo a cui bisogna dare più importanza;
- **<button>** permette l'inserimento di un bottone cliccabile, nel progetto usato varie volte principalmente per sottoscrivere i vari form;
- **<a>**, viene utilizzato per creare un link cliccabile, usato dall'utente per reindirizzarsi all'interno delle pagine web;

- `<table>` tag che permette di stampare tabelle, al suo interno sono presenti altri tag: `<tr>` per le colonne, `<td>` per le righe e `<tr>` per le intestazioni. L'uso che è stato fatto nelle varie blade dell'applicativo è quello di rappresentare le varie tabelle da modificare del database e mostrare il risultato delle schede di valutazione. [13]

Capitolo 3

Ingegnerizzazione del software e lo schema architetturale proposto

Dopo aver spiegato le principali tecnologie per sviluppare questo progetto, qui di seguito verranno spiegate in modo dettagliato le funzioni del servizio, le pagine web che lo compongono e la logica dietro al suo funzionamento.

Il progetto tratta la creazione di un applicativo web il cui scopo principale è quello di creare un servizio capace di svolgere l'attività di censimento e valutazione del trattamento dei dati nel modo più intuitivo possibile.

L'applicativo è stato sviluppato in modo da permettere l'accesso ai tre soggetti competenti:

- utente **Admin**, l'utente di livello più alto, lui ha pieni poteri sull'applicativo come anche la gestione del database.
- utente **capo struttura**, che corrisponde alla figura del Direttore di Dipartimento, il quale ha capacità di modifica sul suo dipartimento e di lettura sugli altri soggetti allo stesso livello;

- utente **delegato**, nominato dal capo struttura, le sue funzioni riguardano solo la compilazione e visualizzazione dei form.

Il progetto riguarda la creazione di un applicativo che ha lo scopo di fornire un panorama sul trattamento dei dati personali svolto all'interno di una struttura universitaria da parte degli utenti che usufruiranno di un servizio. Gli utenti una volta effettuata la registrazione e l'accesso, potranno svolgere diverse funzioni all'interno dell'applicativo in base al loro ruolo, in particolare, i capi struttura potranno nominare delegati, modificare delle tabelle del database, creare una nuova scheda di valutazione del trattamento e visualizzare, modificare una lista di schede di valutazioni che si riferiscono al loro dipartimento o struttura associato. Mentre i delegati avranno solo il ruolo di creare nuove schede di valutazione per la struttura assegnata.

Il prodotto finale del servizio è quello di fornire un quadro in forma tabellare che mostri una rappresentazione sommaria delle caratteristiche dei dati in possesso della struttura scelta, mostrando gli indici di rischio correlati ad ogni singolo ambito. In fondo allo schema completato si troveranno gli indici di rischio, calcolati in base ai dati inseriti in fase di scrittura della scheda di valutazione e riguardanti la gravità dei valori R, I e D (Riservatezza, Integrità e Disponibilità).

Digitando su un qualsiasi motore di ricerca, l'indirizzo "gdpregistry.iccsa.org", ci troveremo davanti ad una pagina, chiamata `welcome.blade.php`, che è la prima pagina che viene visualizzata dall'utente. La pagina è composta dal logo e dal nome, due bottoni per fare la registrazione e l'accesso (`register`, `login`), una vista generica con in riferimento dei link utili legati al tema del progetto ed un collegamento verso le schede e i form accessibili agli utenti

The image shows a registration form titled "Registrati". It contains the following fields and controls:

- Nome: Text input field
- Cognome: Text input field
- Lingua: Dropdown menu with a downward arrow
- Dipartimento: Dropdown menu with a downward arrow
- Indirizzo Email: Text input field
- Password: Text input field
- Conferma Password: Text input field
- Registrati: Blue button with white text

Figura 3.1: Registrazione

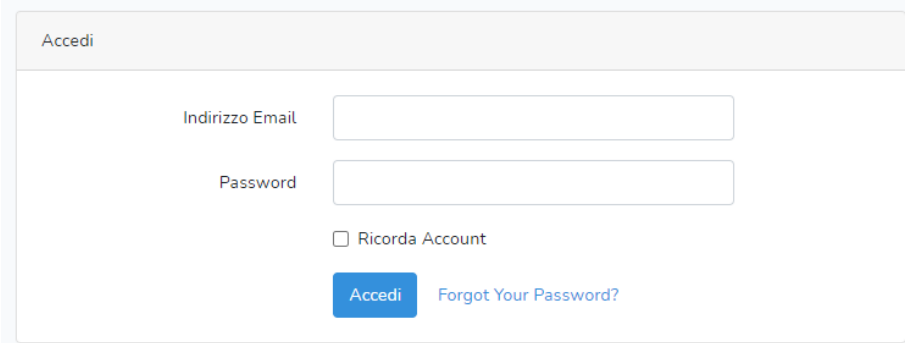
ospiti.

Uno degli strumenti di Laravel è quello di facilitare la scrittura delle pagine che hanno tra loro elementi in comune, ad esempio il menù utente e la navbar sono presenti in tutte le altre pagine estendendo, grazie al comando `@extends('layouts.app')`, gli elementi presenti nel blade `app.blade.php` in cui si trovano anche i riferimenti stilistici.

In questa pagina principale, grazie ai bottoni presenti, sarà possibile passare alle due pagine per la registrazione e l'accesso.

3.1 Registrazione

Questa pagina è composta da un form (figura 3.1) strutturato in sette etichette e sette caselle di input dove l'utente dovrà inserire i propri dati, tra



The image shows a login form titled "Accedi". It contains two input fields: "Indirizzo Email" and "Password". Below the "Password" field is a checkbox labeled "Ricorda Account". At the bottom of the form are two buttons: a blue "Accedi" button and a blue "Forgot Your Password?" link.

Figura 3.2: Login

cui la lingua, il dipartimento di appartenenza, nome e cognome dell'utente e la conferma della password. Per i campi lingua e dipartimento sono stati predisposti dei menù a tendina per costringere ad una risposta standard. I dati inseriti dall'utente vengono convalidati dalla funzione `validate()` che oltre alla normale verifica dei dati controlla anche la validità della mail, che deve rispettare il formato `nome@dominio.xxx` e la coincidenza delle password inserite.

3.2 Login

Strutturata come la pagina di registrazione, ma con la differenza di contenere due etichette e due caselle di input invece che sette, dove l'utente già registrato dovrà inserire la propria mail e la propria password (figura 3.2). Anche in questo caso le credenziali inserite dall'utente vengono convalidate dall'apposita funzione che confronterà i dati inseriti e quelli presenti nel database, consentendo l'accesso se la verifica ha avuto esito positivo altrimenti dando un messaggio di errore.

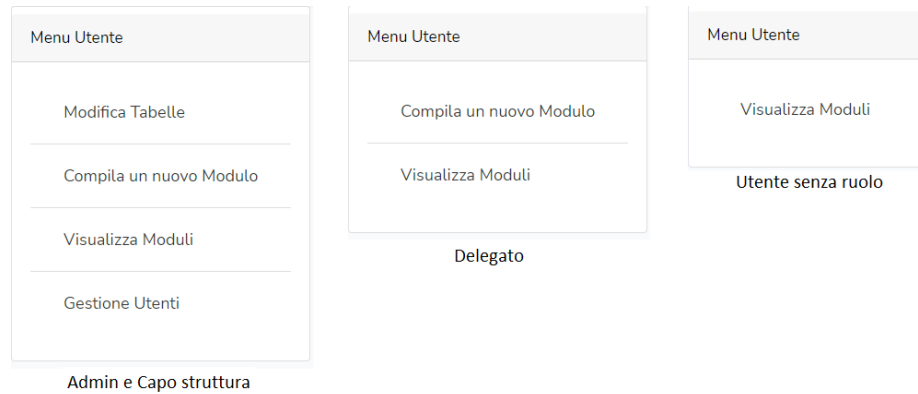


Figura 3.3: Diversi menu utente

3.3 Pagina Home

L'utente, una volta registrato o dopo essere acceduto, si ritrova nella pagina `home_c.blade.php`, dove in alto è presente la barra di navigazione in cui si trovano due menu a tendina, uno per selezionare la lingua e uno con il collegamento alla pagina 'profilo' e per il collegamento per fare il logout. Mentre sulla sinistra è presente il menu utente, il quale a seconda del tipo di utente avrà voci diverse (figura 3.3).

Avremo quindi disponibili quattro possibili funzioni da scegliere sulla sinistra:

- **Modifica tabelle**, disponibili ai capistruttura e agli utenti di primo livello.
- **Compila un nuovo Modulo**, per tutti gli utenti con profilo attivato;
- **Visualizza moduli**, per tutti gli utenti ma con restrizioni sulle modifiche e visualizzazioni;



Figura 3.4: Sezione di edit.blade.php

- **Gestione utenti**, utilizzabile solo per gli Admin nella totalità delle sue funzioni, mentre è disponibile anche agli utenti intermedi ma con funzioni limitate;

3.4 Modifica tabelle

Dopo aver premuto il link corrispondente alla prima voce del menu utente, si entrerà nella pagina `edit.blade.php`, la quale è composta da una finestra centrale dove tramite un costrutto `foreach` verranno stampate tutte le tabelle modificabili (figura 3.4), cioè le tabelle contenenti le voci da inserire nei form di valutazione. Come vedremo nel paragrafo 4.2.2, questo elenco viene preso dalla tabella `Types_of_data` che funge da filtro, permettendo di dividere i vari record in base al tipo di dato.



Form di inserimento dati con il titolo "Inserisci Riga". Il form contiene i seguenti campi:

- description_it: campo di testo.
- description_en: campo di testo.
- R: menu a tendina.
- I: menu a tendina.
- D: menu a tendina.
- automated_process: menu a tendina.
- data_banks: menu a tendina.
- Aggiungi: pulsante verde per salvare il record.

Figura 3.5: Form di inserimento

Ognuna delle righe stampate dal ciclo, avrà come campi il nome della tabella e tre bottoni, i quali danno la possibilità di inserire un nuovo record, di modificarlo o anche cancellarlo.

3.4.1 Inserisci record

Se l'utente vuole inserire una nuova riga all'interno di una tabella allora basterà premere il relativo bottone verde, questo dalla tabella precedente si salverà la chiave primaria, corrispondente al tipo di dati, necessaria per identificare la tabella da modificare. Questo parametro verrà inviato

Scegli una riga da eliminare

treatment_categories

description_it	description_en	R	I	D	automated_process	data_banks	state	
Gestione standard del ciclo di vita dei dati		1.1	1.1	1.1	NO	NO	ATTIVO	<button>Cancella</button>
Collezione		1	1	1	NO	NO	ATTIVO	<button>Cancella</button>
Registrazione		1	1.1	1	NO	NO	ATTIVO	<button>Cancella</button>
Archiviazione		1.1	1.1	1.1	NO	NO	ATTIVO	<button>Cancella</button>
Uso		1.1	1	1	NO	NO	ATTIVO	<button>Cancella</button>
Aggiornamento		1	1	1	NO	NO	ATTIVO	<button>Cancella</button>

Figura 3.6: delete_t.blade

al controller relativo `Insert_TController.php` che dopo aver selezionato la tabella corrispondente passerà una serie di variabili necessarie per stampare i menu a tendina alla pagina successiva, dove comparirà il form di inserimento per una nuova riga (figura 3.5).

Una volta che l'utente preme il bottone 'Aggiungi', i dati inseriti nel form e il nome dello schema vengono inviati tramite il metodo POST al controller designato per l'inserimento dei campi, nella fattispecie la funzione `store()` nel file `TableListController` verificherà i campi inseriti validandoli. Se l'inserimento va a buon fine si verrà rimandati sulla pagina che lista tutte le tabelle da modificare.

3.4.2 Elimina record

Premendo il tasto rosso, si verrà renderizzati nella pagina `delete_t.blade`, la quale avrà la stessa logica per la visualizzazione dei campi, ma nell'ultima cella sarà presente un bottone (come in figura 3.6) che una volta premuto

age_range_of_the_data_subjects ROW 3		Modifica i campi di : age_range_of_the_data_subjects
<div style="margin-bottom: 10px;"> range Adults and / or children under 18 (> 14 years) </div> <div style="display: flex; justify-content: space-between;"> <div style="width: 40%;"> R 1.1 </div> <div style="width: 40%;"> I 1.1 </div> <div style="width: 40%;"> D 1.1 </div> </div> <div style="margin-top: 10px;"> change_id </div>		<div style="margin-bottom: 10px;">ATTENZIONE: i campi uguali devono essere riscritti</div> <div style="margin-bottom: 10px;"> range <input style="width: 100%;" type="text"/> </div> <div style="display: flex; justify-content: space-around; margin-bottom: 10px;"> <div style="text-align: center;"> R <input style="width: 100%;" type="text"/> </div> <div style="text-align: center;"> I <input style="width: 100%;" type="text"/> </div> <div style="text-align: center;"> D <input style="width: 100%;" type="text"/> </div> </div> <div style="text-align: center;"> <input style="background-color: #4a7ebb; color: white; padding: 5px 10px;" type="button" value="change"/> </div>

Figura 3.7: Form di aggiornamento

azionerà la funzione nel relativo controller per eliminare il giusto record.

3.4.3 Modifica record

La pagina per la modifica di un campo è stata strutturata con lo stesso principio dell'eliminazione, ma al posto del tasto 'cancella' è presente il tasto 'modifica', il quale reindirizza all'interno della pagina `update.blade.php`. Questa è costruita con due finestre affiancate (come in figura 3.7), una con i dati della riga che si vuole modificare mentre nell'altra si trova il modulo per cambiare i dati.

Anche in questo caso i dati verranno inviati alla giusta funzione per essere validati, nella fattispecie alla funzione `update()` in `TableListController.php`.

Listing 3.1: Funzione di update in TableListController.php

```

1 public function update(Request $request , $table , $id){
2     $change_id= Auth::user()->id;
  
```

```
3      $type = DB::table('types_of_data')->select('id')->
      where('name', '=', $table)->get();
4      if($table == 'assets_physical_archives'){
5          $request->validate([
6              'archives_it' => 'required|max:255',
7              'archives_en' => 'required|max:255',
8              'synthetic_classification' => 'required|
              max:10',
9              'process_fk' => 'required|max:10'
10         ]);
11         DB::table('assets_physical_archives')->where('
            id',$id)->update(['change_id'=>$change_id,
            'state'=>0]);
12         assets_physical_archives::create([
13             'type' => $type[0]->id,
14             'archives_it' =>$request->archives_it,
15             'archives_en' =>$request->archives_en,
16             'synthetic_classification'=>$request->
                synthetic_classification,
17             'process_fk'=>$request->process_fk,
18             'change_id' => $change_id,
19             'state' => 1
20         ]);
21         return Redirect::back()->with('completed', '
            table_has_been_saved');
```

Parte 1: Intestazione

I campi contrassegnati da un asterisco (*) sono obbligatori

* Titolare

* Tassonomia del servizio

* Denominazione sintetica del trattamento

Prossimo

Figura 3.8: Prima parte del flusso

In questa parte di codice si può vedere la funzione `validate()`. Una volta validati i campi, il record modificato viene aggiornato settando lo stato a '0' cioè diventerà un record non attivo (come vediamo a riga 11), mentre i nuovi dati inseriti andranno a creare una nuova riga della tabella.

3.5 Schede di trattamento

Premendo la voce 'Compila un nuovo Modulo' l'utente verrà reindirizzato nella pagina `fillForm.blade.php` contenente le prime voci da riempire.

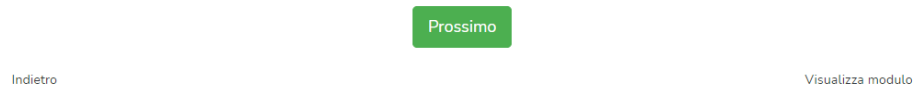


Figura 3.9: Bottoni sotto ai form

re della scheda di valutazione (vedi figura 3.8), nel mentre avverranno in sequenza una serie di azioni svolte dalla funzione `index()` nel controller `fillFormController`:

- verrà fatto un controllo sul ruolo dell'utente
- verrà creato un seriale randomico che verrà utilizzato come chiave univoca per identificare i vari form;
- all'interno della tabella `form` saranno inseriti l'id dell'utente, il Dipartimento e la struttura di riferimento, un timestamp per memorizzare la data di inizio trattamento ed infine la chiave univoca del secondo punto.

Quello mostrato in figura 3.8 è il primo di una serie di nove form che andranno a riempire la tabella `form_answers`. Ognuno di questi moduli compilabili oltre ai campi da inserire ha, nella parte bassa della finestra, tre link, mostrati nella figura 3.9.

Premendo il bottone che farà proseguire la scrittura dei moduli sarà azionata una funzione che come prima cosa valida i dati inseriti poi svolge un controllo sul database andando a verificare che non esistano già all'interno della tabella `form_answers` delle righe corrispondenti alla prima parte del flusso con il seriale uguale a quello che si sta utilizzando per questa sessione, se così fosse elimina quelle righe e inserisce al loro posto quelle nuove.

Gli altri due pulsanti sono, intuitivamente, per la visualizzazione del form incompleto e la possibilità di tornare indietro e modificare i campi inseriti.

Arrivati a compilare tutti i form, ci si troverà davanti ad un'ultima finestra di conferma che presenterà il collegamento alla funzione `completeForm()` del controller `fillFormController` la quale, in base ai dati inseriti dall'utente in fase di compilazione della scheda di valutazione, andrà a calcolare ed inserire nella relativa tabella i vari indici di rischio, e inoltre reindirizzerà l'utente nella pagina `completeForm`, dove sarà visualizzato lo schema della scheda completata.

3.5.1 Visualizzazione schede

Selezionando la voce del menu si entrerà nella pagina che mostra i moduli (vedi figura 3.10). Verranno mostrate due finestre con relative tabelle una con i moduli completi e una con i moduli incompleti, a seconda del ruolo dell'utente sarà possibile visualizzarne e modificarne una parte piuttosto che tutte quante.

Le 3 gerarchie di utenti potranno quindi visualizzare, modificare o eliminare una scheda di trattamento dei dati seguendo le seguenti regole:

- Un utente di basso livello può visualizzare solo i form compilati nel suo Dipartimento ed eliminare o aggiornare solo quelli compilati da lui stesso.
- Un utente di livello intermedio può visualizzare le proprie schede di valutazione e quelle degli altri Dipartimenti, ha però potere di verifica e cancellazione solo su quelli del proprio Dipartimento.

Ingegnerizzazione del software e lo schema architetturale proposto

COMPLETED FORMS							
id	department	holder	employee	structure	evaluation date	form serial	view
42	Mathematics and Computer Science		31		2021-01-27 16:06:41	serial1608821807	VIEW
45	Chemistry, biology and biotechnology		31		2021-01-27 16:06:44	serial1608980281	VIEW
80	Mathematics and Computer Science		31		2021-01-27 16:06:47	serial1610622306	VIEW
91	Mathematics and Computer Science		41		2021-01-27 21:40:05	serial1611755636	VIEW

EDIT FORM							
id	department	holder	employee	structure	evaluation date	form serial	view
80	Mathematics and Computer Science		31		2021-01-27 16:06:47	serial1610622306	EDIT DELETE

Figura 3.10: Visualizzazione delle tabelle dei form

Ingegnerizzazione del software e lo schema architetturale proposto

SCHEMA	
HEADING	
HOLDER	Università degli Studi di Perugia
SERVICE TAXINOMY	Responsabile del servizio di sicurezza e prevenzione
SHORT DEN	trattamenti dati personali e sanitari Covid19
edit	

Figura 3.11: Esempio di presentazione della scheda in modalità aggiornamento

- L'utente di livello più alto avrà pieni poteri di gestione su tutte le schede di valutazione compilate di tutti i Dipartimenti.

Se l'utente seleziona il bottone relativo alla visualizzazione o modifica del form, potrà visionarlo in un'altra pagina. Nel secondo caso, sotto le varie sezioni compilate (o meno) si avrà a disposizione il pulsante per la modifica di una delle sezioni (vedi figura 3.11). Questo farà partire un metodo che porterà al form ad esso relativo consentendo di ricompilarne i dati.

Capitolo 4

Descrizione dell'applicazione realizzata

4.1 Backend: ambiente e funzioni

Un ambiente Backend è la parte del programma che permette il corretto funzionamento delle interazioni che vi sono tra gli utenti ed il corrispettivo ambiente Frontend, in modo che tutte le operazioni avvengano senza errori di nessun tipo.

La differenza sostanziale con il Frontend, è che questo identifica la parte del sistema visibile agli utenti con cui essi possono interagire, ovvero un' interfaccia utente. Quindi l'accesso in quest'ultimo caso è riservato agli utenti finali, mentre per il Backend, l'accesso è permesso soltanto agli amministratori del sito che possono effettuare diverse operazioni, tra le quali troviamo:

- creazione di nuove pagine web da aggiungere al sito e modifica di

pagine web già esistenti;

- gestione dell'aspetto grafico;
- attivazione/disattivazione di alcune funzionalità;

4.2 Il database MySQL

Per la raccolta dei dati e la loro gestione, è stato utilizzato un database gestito tramite MySQL.

Il termine Database indica una struttura organizzata di dati omogenea sia nel formato che nel contenuto. Sono considerati fondamentali per i moderni siti web dinamici e sono indicati come DBMS (Database Management System) ed il loro utilizzo principale è quello di memorizzare delle informazioni, per essere poi utilizzate in diversi siti, programmi o applicazioni. Per gestire un Database, possiamo effettuare diverse operazioni come inserire, eliminare, visualizzare o modificare i dati.

Esistono diversi Software per la gestione dei DBMS , tra questi vi è MySQL, che è un Relational Database Management System (RDBMS) open source, sotto il controllo della Oracle, ed è distribuito in versione gratuita con licenza General Public License (GNU). Deriva da SQL, linguaggio che permette elaborazioni rapide con affidabilità, flessibilità e facilità di utilizzo.

MySQL è supportato da diversi linguaggi di programmazione, come ad esempio PHP e Java e rappresenta una componente fondamentale per sviluppare applicazioni open source. [15]

4.2.1 istruzioni SQL

Per poter gestire al meglio un database, è necessario conoscere le 4 operazioni base messe a disposizione da SQL :

- **INSERT**: l'operazione utilizzata per inserire nuovi dati all'interno del database, la sua sintassi di base è `"INSERT INTO nome_tabella(colonna1, colonna2, ecc..) VALUES (valore1, valore2, ecc..)"`, dove i valori verranno inseriti nella corrispettiva colonna della tabella, nell'ordine indicato;
- **SELECT**: E' utilizzata per estrapolare informazioni dal database. La sua sintassi di base è `"SELECT * FROM nome_tabella"`, dove con `'*'` si indica che saranno prese dalla tabella, tutte le colonne, ovvero tutte le informazioni. E' possibile diminuire il numero di dati da prendere (nel caso non ci servissero tutti), scrivendo al posto dell'asterisco, il nome della/e collona/e da prendere in considerazione.
- **UPDATE**: utilizzata per aggiornare la struttura o i valori di una tabella o una singola colonna, la sua sintassi di base è `"UPDATE nome_tabella SET nome_colonna = valore"`, con **SET** si imposta il nuovo valore della colonna indicata (E' possibile indicare più di una colonna da modificare).
- **DELETE**: utilizzata per cancellare le righe di una tabella, la sua sintassi di base è `"DELETE FROM nome_tabella"` , in questo caso verranno eliminate tutte le righe della tabella.

Inoltre è possibile restringere il campo di effetto di una certa istruzione ad un determinato numero di colonne. A questo scopo si utilizza **WHERE**

nelle istruzioni viste in precedenza (esclusa quella di inserimento). Con questa istruzione, si indica che saranno prese in considerazione, soltanto le righe che soddisfaranno il predicato indicato dopo WHERE.

4.2.2 Struttura del database

Per gestire al meglio il sito, sono state create diverse tabelle che fanno parte di uno stesso database (vedi figura 4.1). Le tabelle, per far comprendere al meglio la struttura, verranno divise in tre macro gruppi: quelle utili per la gestione degli utenti, quelle contenenti i dati da inserire nei form e infine quelle per la gestione dei form stessi.

Tabelle per la gestione degli utenti

Users

```
CREATE TABLE 'users' (  
    'id' int(10) NOT NULL, /*chiave primaria*/  
    'name' varchar(100) NOT NULL,  
    'surname' varchar(100) NOT NULL,  
    'language_id' int(10) NOT NULL, /*chiave esterna  
su tab. 'language'*/  
    'role' int(10) DEFAULT 3, /*chiave esterna  
su tab. 'roles'*/  
    'structure' int(10) DEFAULT NULL, /*chiave esterna  
su tab. 'structures'*/  
    'department_id' int(10) NOT NULL, /*chiave esterna  
su tab. 'departments'*/
```

Descrizione tecnica dell'applicazione realizzata

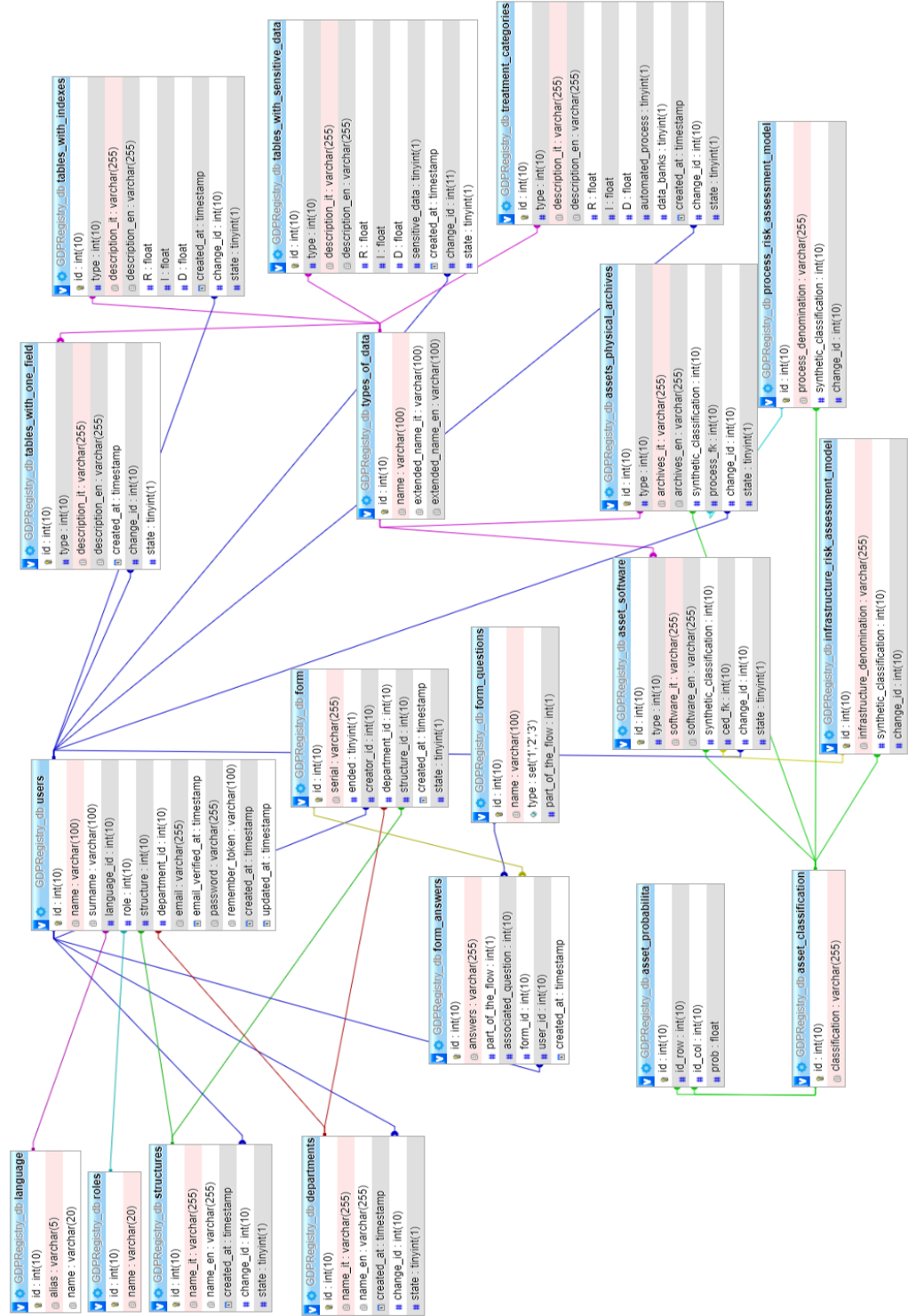


Figura 4.1: Schema E/R del database


```
'email' varchar(255) NOT NULL,  
'email_verified_at' timestamp NOT NULL DEFAULT current_timestamp(),  
'password' varchar(255) NOT NULL,  
'remember_token' varchar(100) DEFAULT NULL,  
'created_at' timestamp NULL DEFAULT NULL,  
'updated_at' timestamp NULL DEFAULT NULL  
);
```

Roles

```
CREATE TABLE 'roles' (  
  'id' int(10) NOT NULL,  
  'name' varchar(20) NOT NULL  
);
```

Language

```
CREATE TABLE 'language' (  
  'id' int(10) NOT NULL,  
  'alias' varchar(5) NOT NULL,  
  'name' varchar(20) NOT NULL  
);
```

Departments

```
CREATE TABLE 'departments' (  
  'id' int(10) NOT NULL,  
  'name_it' varchar(255) NOT NULL,  
  'name_en' varchar(255) DEFAULT NULL,  
  'created_at' timestamp NOT NULL DEFAULT current_timestamp(),
```

```
'change_id' int(10) DEFAULT NULL COMMENT,  
'state' tinyint(1) NOT NULL DEFAULT 1  
);
```

Structures

```
CREATE TABLE 'structures' (  
  'id' int(10) NOT NULL,  
  'name_it' varchar(255) NOT NULL,  
  'name_en' varchar(255) DEFAULT NULL,  
  'created_at' timestamp NOT NULL DEFAULT current_timestamp(),  
  'change_id' int(10) DEFAULT NULL,  
  'state' tinyint(1) NOT NULL DEFAULT 1  
);
```

Le tabelle precedenti raccolgono informazione utili per l'identificazione e gestione degli utenti che avranno accesso al sito. Ogni record della tabella **'users'** corrisponde ad un utente iscritto alla piattaforma, i dati inseriti nel form di registrazione (vedi nella sezione 3.1) andranno salvati direttamente qua. I campi **'language_id'**, **'role'**, **'structure'** e **'department_id'** sono chiavi esterne, rispettivamente, sulle tabelle: **'language'**, **'roles'**, **'structures'** e **'departments'**.

Tabelle contenenti i dati

Le tabelle descritte in questo paragrafo, contengono tutti i valori che serviranno all'utente per compilare una scheda di valutazione. Tutte tranne la tabella **'Types_of_data'** che funge da filtro, infatti tutte le altre hanno un campo in comune **'type'**, utile per identificare il tipo di dato, il quale

è chiave esterna proprio sul campo 'id' della tabella 'Types_of_data'. Le tabelle elencate sotto inoltre hanno un altro campo in comune 'change_id' che servirà per tenere traccia di chi ha modificato un certo record.

Types_of_data

```
CREATE TABLE 'types_of_data' (  
    'id' int(10) NOT NULL,  
    'name' varchar(100) NOT NULL,  
    'extended_name_it' varchar(100) NOT NULL,  
    'extended_name_en' varchar(100) NOT NULL  
);
```

Tables_with_one_field

```
CREATE TABLE 'tables_with_one_field' (  
    'id' int(10) NOT NULL,  
    'type' int(10) NOT NULL, /*chiave esterna  
su tab. 'types_of_data'*/  
    'description_it' varchar(255) NOT NULL,  
    'description_en' varchar(255) DEFAULT NULL,  
    'created_at' timestamp NOT NULL DEFAULT current_timestamp(),  
    'change_id' int(10) DEFAULT NULL, /*chiave esterna  
su tab. 'users'*/  
    'state' tinyint(1) NOT NULL DEFAULT 1  
);
```

Tables_with_indexes

```
CREATE TABLE 'tables_with_indexes' (  
    'id' int(10) NOT NULL,
```

```
'type' int(10) NOT NULL, /*chiave esterna
su tab. 'types_of_data'*/
'description_it' varchar(255) NOT NULL,
'description_en' varchar(255) DEFAULT NULL,
'R' float NOT NULL,
'I' float NOT NULL,
'D' float NOT NULL,
'created_at' timestamp NOT NULL DEFAULT current_timestamp(),
'change_id' int(10) DEFAULT NULL, /*chiave esterna
su tab. 'users'*/
'state' tinyint(1) NOT NULL DEFAULT 1
);
```

Tables_with_sensitive_data

```
CREATE TABLE 'tables_with_sensitive_data' (
'id' int(10) NOT NULL,
'type' int(10) NOT NULL, /*chiave esterna
su tab. 'types_of_data'*/
'description_it' varchar(255) NOT NULL,
'description_en' varchar(255) DEFAULT NULL,
'R' float NOT NULL,
'I' float NOT NULL,
'D' float NOT NULL,
'sensitive_data' tinyint(1) NOT NULL,
'created_at' timestamp NOT NULL DEFAULT current_timestamp(),
'change_id' int(11) DEFAULT NULL, /*chiave esterna
```

```
su tab. 'users'*/  
'state' tinyint(1) NOT NULL DEFAULT 1  
);
```

Treatment_categories

```
CREATE TABLE 'treatment_categories' (  
  'id' int(10) NOT NULL,  
  'type' int(10) NOT NULL DEFAULT 1,/*chiave esterna  
su tab. 'types_of_data'*/  
  'description_it' varchar(255) NOT NULL,  
  'description_en' varchar(255) DEFAULT NULL,  
  'R' float NOT NULL,  
  'I' float NOT NULL,  
  'D' float NOT NULL,  
  'automated_process' tinyint(1) NOT NULL COMMENT,  
  'data_banks' tinyint(1) NOT NULL COMMENT ,  
  'created_at' timestamp NOT NULL DEFAULT current_timestamp(),  
  'change_id' int(10) DEFAULT NULL,/*chiave esterna  
su tab. 'users'*/  
  'state' tinyint(1) NOT NULL DEFAULT 1  
);
```

Asset_software

```
CREATE TABLE 'asset_software' (  
  'id' int(10) NOT NULL,  
  'type' int(10) NOT NULL DEFAULT 16,/*chiave esterna  
su tab. 'types_of_data'*/
```

```
'software_it' varchar(255) NOT NULL,  
'software_en' varchar(255) DEFAULT NULL,  
'synthetic_classification' int(10) NOT NULL COMMENT,  
'ced_fk' int(10) NOT NULL,  
'change_id' int(10) DEFAULT NULL, /*chiave esterna  
su tab. 'users' */  
'state' tinyint(1) NOT NULL DEFAULT 1  
);
```

Assets_physical_archives

```
CREATE TABLE 'assets_physical_archives' (  
  'id' int(10) NOT NULL,  
  'type' int(10) NOT NULL DEFAULT 17, /*chiave esterna  
su tab. 'types_of_data' */  
  'archives_it' varchar(255) NOT NULL,  
  'archives_en' varchar(255) DEFAULT NULL,  
  'synthetic_classification' int(10) NOT NULL COMMENT,  
  'process_fk' int(10) NOT NULL,  
  'change_id' int(10) DEFAULT NULL, /*chiave esterna  
su tab. 'users' */  
  'state' tinyint(1) NOT NULL DEFAULT 1  
);
```

Tabelle per la gestione dei form

Infine troviamo le tabelle per la gestione dei form. Ogni record della prima tabella corrisponde ad una scheda di valutazione creata. La tabella

'Form_questions' contiene tutte le domande, obbligatorie o non obbligatorie, a cui l'addetto dovrà rispondere per il completamento della scheda. I record dell'ultima tabella contengono le risposte date, 'Form_questions' ha come chiave esterna sulla seconda tabella il campo 'associated_question' che tiene traccia a quale domanda corrisponde tale risposta.

Form

```
CREATE TABLE 'form' (  
  'id' int(10) NOT NULL,  
  'serial' varchar(255) NOT NULL,  
  'ended' tinyint(1) NOT NULL,  
  'creator_id' int(10) NOT NULL, /*chiave esterna  
  su tab. 'users'*/  
  'department_id' int(10) NOT NULL,  
  'structure_id' int(10) NOT NULL,  
  'created_at' timestamp NOT NULL DEFAULT current_timestamp(),  
  'state' tinyint(1) NOT NULL  
);
```

Form_questions

```
CREATE TABLE 'form_questions' (  
  'id' int(10) NOT NULL,  
  'name' varchar(100) NOT NULL,  
  'type' set('1','2','3') NOT NULL,  
  'part_of_the_flow' int(1) NOT NULL  
);
```

Form_answers

```
CREATE TABLE 'form_answers' (  
    'id' int(10) NOT NULL,  
    'answers' varchar(255) DEFAULT NULL,  
    'part_of_the_flow' int(1) NOT NULL,  
    'associated_question' int(10) NOT NULL, /*chiave esterna  
    su tab. 'form_questions' */  
    'form_id' int(10) NOT NULL,  
    'user_id' int(10) NOT NULL, /*chiave esterna  
    su tab. 'users' */  
    'created_at' timestamp NOT NULL DEFAULT current_timestamp()  
);
```

4.2.3 gestione del database tramite Laravel

Per gestire le operazioni da effettuare sul database, sono stati usati vari file all'interno del progetto Laravel.

Prima di tutto il database dovrà essere configurato. Il file di configurazione dedicato al database è `config/database.php` ma il miglior modo di configurare i parametri è utilizzare il file `.env` in modo da poter riutilizzare gli stessi file su diversi ambienti; `.env` contiene infatti tutte le configurazioni dipendenti dall'ambiente. Le property interessate sono:

- `DB_CONNECTION`, contiene la tipologia di database utilizzato;
- `DB_HOST`, contiene l'ip o l'indirizzo del server db;
- `DB_PORT`, contiene la porta del db;
- `DB_DATABASE`, contiene il nome del database;

- `DB_USERNAME`, contiene l'username;
- `DB_PASSWORD`, contiene la password.

Dopo aver configurato la connessione al database, è possibile effettuare operazioni sul db utilizzando il Query Builder, il quale fornisce un'interfaccia comoda e fluida per creare ed eseguire query. Il generatore di query Laravel utilizza l'associazione dei parametri PDO per proteggere l'applicazione dagli attacchi di SQL injection, perciò non sarà necessario controllare o pulire le stringhe passate al Query Builder. [16]

Esecuzione delle query

È possibile utilizzare il metodo `table` fornito dalla facade DB per iniziare una query. Il metodo `table` restituisce un'istanza fluida del Query Builder per la tabella data, consentendo di concatenare più vincoli sulla query. [16]

Nel progetto, lo strumento fornito da Laravel è stato usato su vari punti, tra questi:

Listing 4.1: SELECT

```
1 $languages = DB::table('language')->select('id', 'name')->get();
```

select semplice usata nel `RegisterController` per passare alla vista i tipi di lingua che l'utente dovrà scegliere in fase di registrazione;

Listing 4.2: UNION

```
1 $q11= DB::table('treatment_categories')->select('id', 'type', 'description_it', 'description_en')->where('state', '=', 1);
```

```

2 $ql2 = DB::table('tables_with_indexes')->select('id',
    'type', 'description_it', 'description_en')->
    whereBetween('type', [2, 4])->where('state', '=',
    1);
3 $tabtot = DB::table('tables_with_one_field')->select('
    id', 'type', 'description_it', 'description_en')->
    whereBetween('type', [5,7])->where('state', '=', 1)
    ->union($ql1)->union($ql2)->get();

```

Concatenazione di query con il metodo **UNION**, il quale permette di unire due o più query insieme. Usato nel file `fillFormController` nella funzione `show()`. Utile per passare le risposte alle domande presenti nella parte 3 della compilazione delle schede di valutazione. Questo metodo permetterà di far svolgere al database una sola query invece che tre separate, molto utile per ottimizzare la velocità nell'interrogare la base di dati. Inoltre viene usato anche il metodo **WHERE** in due forme, con la clausola semplice **where** e **whereBetween**, che come detto sopra serviranno per prendere solo i campi che soddisfano le clausole.

Listing 4.3: INSERT

```

1 DB::table('form_answers')->insert([
2     ['answers' => $request->purpose, 'part_of_the_flow'
        => 2, 'associated_question' => 13, 'form_id'
        => $form_id[0]->id, 'user_id'=> Auth::user()->
        id, 'created_at' => date('Y-m-d_H:i:s')],
3     ['answers' => $request->holder, 'part_of_the_flow'
        => 2, 'associated_question' => 14, 'form_id'=>

```

```
4      $form_id[0]->id , 'user_id'=> Auth::user()->id ,  
      'created_at' => date( 'Y-m-d_H:i:s' ) ] ,  
      .....  
    ] ) ;
```

Il Query Builder fornisce anche un metodo **INSERT** che può essere utilizzato per inserire record nella tabella del database. In questo caso usato nel file `fillFormController` nella funzione `store()`, per inserire le risposte date nella parte 3 della scheda di valutazione, dopo essere stati validati.

Listing 4.4: UPDATE

```
1 DB::table( 'users ')->where( 'id' , '=' , $id)->update( [  
2     'department_id' => $dep ,  
3     'name' => $name ,  
4     'surname' => $surname ,  
5     'email' => $email ,  
6     'structure' => $str ,  
7     'role' => $role  
8 ] ) ;
```

Oltre a inserire i record nel database, il generatore di query può anche aggiornare i record esistenti utilizzando il metodo **UPDATE**. Il metodo **UPDATE**, come il metodo **INSERT**, accetta un array di colonne e coppie di valori che indicano le colonne da aggiornare. Questo metodo è stato usato nel file `DemoController` nella funzione `update()`, richiamata dalla vista `editUser` quando l'Admin modifica i dati di un utente.

Listing 4.5: DELETE

```
1 DB::table('form_answers')->where('form_id', '=',  
    $form_id[0]->id)->where('part_of_the_flow', '=',1)  
->delete();
```

Il metodo **DELETE** può essere utilizzato per eliminare i record della tabella. Inoltre si possono aggiungere vincoli, come mostrato nel codice 4.5, aggiungendo clausole **where** prima di chiamare il metodo **DELETE**.

Conclusioni

Lo scopo di questo progetto è stato quello di creare un'applicazione web in grado di sostituire il vecchio strumento utilizzato dall'ateneo per la compilazione e gestione delle schede di valutazione.

Come analizzato in questo elaborato, i concetti di *privacy* e *salvaguardia dei dati personali* hanno sempre creato dubbi in merito al loro contenuto, perciò è stato fondamentale l'apprendimento delle conoscenze in ambito della regolamentazione dei trattamenti dei dati personali, la loro acquisizione e la gestione da parte delle autorità competenti, al fine di capire l'importanza del programma creato.

Una prima scrittura è stata fatta strutturando l'applicativo in più livelli, questo perché era molto importante che l'applicazione avesse le caratteristiche esatte per gli utenti che l'avrebbero utilizzata, per questo si è svolta con attenzione la suddivisione dei compiti e delle funzioni di ogni singolo soggetto. Questa attenzione non è stata rispecchiata anche per la parte di gestione del database, il che ha portato ad una seconda scrittura dell'applicativo soffermandosi sulla struttura e l'ottimizzazione della sua struttura. La nuova strutturazione ha portato ad una gestione più facile e in maniera efficiente di tutti i dati, ma soprattutto ad una scrittura delle funzioni in PHP per la gestione molto più veloce ed intuitiva.

Prima di questo progetto non mi era mai capitata la necessità di creare un'applicazione web complessa e strutturata, portandomi ad affrontare alcune situazioni particolari. Sebbene i vari linguaggi di programmazione erano già conosciuti per progetti affrontati precedentemente, lo stesso non si può dire di Laravel che si è rilevato essere un framework potente, di facile adozione e con molte componenti interessanti, inoltre la buona documentazione e il buonissimo supporto della comunità favoriscono lo sviluppo di sistemi anche complessi.

Aver svolto questo progetto, sia per le abilità acquisite in ambito di programmazione sia per aver affrontato temi che hanno una posizione fondamentale all'interno dell'ambito informatico, mi saranno molto utili in ambito lavorativo qualora mi fosse richiesto di sviluppare un'altra applicazione simile, facendomi risparmiare tempo in tutte le fasi della progettazione.

Ringraziamenti

Giunti alla conclusione di questo percorso, voglio ringraziare innanzitutto il Prof. Gervasi e i Dott. Perri e Simonetti per la fiducia data per l'assegnazione di questo importante progetto e per l'assistenza nella sua realizzazione.

Desidero ringraziare tutta la mia famiglia per il sostegno e l'incoraggiamento dato durante tutto il percorso universitario, anche quando non proprio aggiornati sull'avanzamento e andamento della mia carriera accademica.

Un ringraziamento particolare va al mio collega Carlo, non solo per il sostegno dato in fase di realizzazione del progetto ma soprattutto per il supporto fornito negli ultimi due anni di studi.

Infine, ringrazio tutti i miei amici vicini e meno vicini, con cui tutti i giorni riesco a passare giornate spensierate.

Bibliografia

- [1] garanteprivacy.it: <https://www.garanteprivacy.it/home/diritti/cosa-intendiamo-per-dati-personali>, visitato in data 27 maggio 2021.
- [2] Corso di formazione in materia di protezione dei dati. Secondo il Regolamento (UE) 2016/679 e il d.lgs. 101/2018. Michele Iaselli, EPC; 1° edizione (26 novembre 2018) 8863108617
- [3] Nicola Fabiano: GDPR & Privacy: consapevolezza e opportunità. L'approccio con il Data Protection and Privacy Relationships Model, goWare, ISBN: 978-8833634074 (29 settembre 2020).
- [4] garanteprivacy.it : <https://www.garanteprivacy.it/home/autorita/compiti>, visitato in data 27 maggio 2021.
- [5] garanteprivacy.it : <https://www.garanteprivacy.it/documents/10160/0/T4DATA+-+Manuale+per+gli+RPD.pdf/bdea3d2d-7bfc-80c5-8f1c-f8c567b44e50?version=1.1>, visitato in data 27 maggio 2021.
- [6] Griffin, Jesse: Introduction to Laravel. ISBN: 978-1-4842-6023-4_4. (2020).

- [7] Matt Stauffer: Laravel: Up & Running: A Framework for Building Modern PHP Apps, O'Reilly Media; 2 edizione (1 aprile 2019) ASIN: B07Q3T513R
- [8] Griffin, Jesse: Advanced Laravel. ISBN: 978-1-4842-6023-4_5. (2020).
- [9] laravel.com:<https://laravel.com/docs/8.x/csrf#csrf-introduction>, visitato in data 27 maggio 2021.
- [10] Kuhn, Darl & Kim, Charles & Lopuz, Bernard: Chapter 12: VirtualBox for Oracle. ISBN: 978-1-4842-1254-7_12. (2015)
- [11] w3schools.com : <https://www.w3schools.com/php/default.asp>, visitato in data 27 maggio 2021.
- [12] Olsson, Mikael: Using PHP. PHP 7 Quick Scripting Reference. Apress, Berkeley, CA, Print ISBN: 978-1-4842-1921-8, Electronic ISBN: 978-1-4842-1922-5, DOI: https://doi.org/10.1007/978-1-4842-1922-5_1(2016).
- [13] w3schools.com : <https://www.w3schools.com/html/default.asp>, visitato in data 27 maggio 2021.
- [14] Rojas, Carlos: Making Your First Web Component, in "Building native Web Components", Apress Editor, Print ISBN: 978-1-4842-5904-7, Electronic ISBN: 978-1-4842-5905-4, DOI: https://doi.org/10.1007/978-1-4842-5905-4_1(2021)
- [15] MYSQL: Guida completa ai database SQL per principianti. Contiene esempi di codice ed esercizi pratici. Oscar R. Frost, Independently published (30 luglio 2020), ASIN : B08DSX8XC6, 979-8670830805

- [16] laravel.com : <https://laravel.com/docs/8.x/queries>, visitato in data 27 maggio 2021.