Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Triennale in
Ingegneria Informatica

Visualizzazione di Dati di medie e piccole aziende

Relatore: Prof. Giorgio Maria Di Nunzio

Laureando: Giacomo Fabbian

Anno Accademico 2018/2019

Indice

1 Introduzione **1**

2 Framework e Modelli Utilizzati **5**
2.1 Modelli 5
2.1.1 Modello Client-Server 5
2.1.2 Ajax Call 6
2.1.3 Pattern MVC 7
2.2 Framework 7
2.2.1 Laravel 7
2.2.2 Google Api Chart 8
2.2.3 jQuery 9
2.2.4 Materialize CSS 10
2.3 Altro 11
2.3.1 CRM 11
2.3.2 ERP 12

3 Descrizione Progetto **13**
3.1 Analisi dei Requisiti 13
3.1.1 Fase 1 13
3.1.2 Fase 2 14
3.2 Progettazione Concettuale 15
3.2.1 Modello Concettuale: Entità-Associazione (E-R) 15
3.2.2 Dizionario dei dati 16
3.3 Software Utilizzati 19
3.3.1 PhpStorm 19
3.3.2 MySQL Workbench 20
3.3.3 Google Chrome 20
3.3.4 Bitbucket 21
3.3.5 Sourcetree 21
3.3.6 Git 22
3.4 Implementazione Progetto 23
3.4.1 PHP Version 23
3.4.2 Laravel Version 24

4 Codice Progetto **25**
4.1 Report JS 25
4.1.1 Ajax Request 25
4.2 Report - PHP version 30
4.2.1 connect.php 30
4.2.2 elabora.php 30
4.3 Report - Laravel Version 32
4.3.1 Risultati ricerca e Grafico 32
4.3.2 Lista nazioni 33
4.3.3 Render Pagina Report 33
4.4 Pagine Lista Dettaglio - Laravel Version 35
4.4.1 Homepage 35
4.4.2 Lista Ordini 35
4.4.3 Dettaglio Ordine 36
4.4.4 Lista Anagrafiche 37
4.4.5 Dettaglio Anagrafiche 38
4.4.6 Lista Nazioni 39
4.4.7 Layout Default 40
4.4.8 Header 41
4.4.9 Footer 43
4.4.10 Comando per generare Aziende Fittizie 44
4.4.11 Comando per generare Lista Attributi 45
4.5 Report SQL 46
4.5.1 Query SQL 46

5 Conclusioni **49**

Elenco delle figure

2.1 Modello Client Server 5
2.2 Ajax Call 6
2.3 Pattern MVC 7
2.4 Logo Laravel 8
2.5 Google Chart API 8
2.6 jQuery 9
2.7 Materialize CSS 10
2.8 CRM 11
2.9 ERP 12

3.1 Schema ER 15
3.2 Php Storm 19
3.3 Mysql Workbench 20
3.4 Google Chrome 20
3.5 Browsers Chart 21
3.6 Bitbucket 21
3.7 SourceTree 22
3.8 Git 22

Elenco dei listati codice

4.1 Ajax Request 25
4.2 connect.php 30
4.3 elabora.php 30
4.4 Risultati ricerca e Grafico 32
4.5 Lista nazioni 33
4.6 Render Pagina Report 33
4.7 Homepage 35
4.8 Lista Ordini 35
4.9 Dettaglio Ordine 36
4.10 Lista Anagrafiche 37
4.11 Dettaglio Anagrafiche 38
4.12 Lista Nazioni 39
4.13 Layout Default 40
4.14 Header 41
4.15 Footer 43
4.16 Comando per generare Aziende Fittizie 44
4.17 Comando per generare Lista Attributi 45
4.18 Query SQL 46

Capitolo 1

Introduzione

In questo lavoro di tesi viene descritta l'esperienza del tirocinio universitario, svolto presso Claim come web developer. Claim è una agenzia pubblicitaria, con sede nella zona del piovese. Questa agenzia non è solamente una web agency; al contrario, è specializzata in tutte le diverse forme di comunicazione, con l'obiettivo di far crescere le aziende clienti nel mondo del web, grazie a molteplici servizi offerti:

- **Advertising, online o tradizionale.** Claim aiuta il cliente a comunicare i valori del marchio, mettendoli insieme in una veste grafica e comunicativa. Concretamente, sviluppa la Brand Identity, creando un logo ed un'impostazione grafica per tutte le comunicazioni del cliente, offline ed online. Tuttavia, non tralascia la comunicazione tradizionale su carta stampata, realizzando la grafica di volantini, biglietti da visita, locandine, brochure e qualsiasi altro documento riguardante il cliente. Claim, negli ultimi anni, ha creato una rete tipografie, litografie e serigrafie per la stampa in qualsiasi formato, in modo esclusivo e dedicato ai propri clienti e ai loro progetti.

- **Siti web, realizzazione di siti web responsive e SEO friendly.** Con l'utilizzo di tecnologie innovative, per il migliore risultato possibile, da qualsiasi tipo di dispositivo e per una maggiore visibilità in rete. L'azienda mette a disposizione di ogni cliente cinque professionisti per ogni fase del progetto, tra cui web designer, web content specialist e front end developer, ma soprattutto un Project Manager di riferimento, per tenere aggiornato il cliente sullo stato di avanzamento del lavoro e per affrontare eventuali problematiche. Claim sviluppa anche e-commerce, dove vendere qualsiasi tipo di prodotto, garantendo usabilità, testi persuasivi e contenuti che portino l'utente ad acquistare.

- **Lead generation.** Attraverso una serie di strategie di email marketing, social media marketing e SEO[1] - SEM[2]. Attraverso la prima strategia, Claim

[1]Search Engine Optimization. Si intendono tutte quelle attività volte a migliorare la scansione, l'indicizzazione e la catalogazione di un documento presente in un sito web, da parte dei crawler dei motori di ricerca

[2]Search engine marketing è il ramo del web marketing che si applica ai motori di ricerca,

cerca di ottimizzare la comunicazione commerciale inviata via e-mail, creando un database per costruire comunicazioni mirate, curare la grafica del messaggio ed, infine, avere a disposizione una reportistica dettagliata, per migliorare costantemente la strategia. Attraverso la strategia di social media marketing, l'azienda gestisce i principali social media, creando contenuti sulle caratteristiche, sulle esigenze e sugli obiettivi che il cliente vuole raggiungere; inoltre, la strategia deve essere diversificata per ogni tipo di social media, creando contenuti ad hoc per ognuno di essi. Anche in questa strategia è importante analizzare continuamente il lavoro attraverso dei report, per poter migliorare. Infine, SEO e SEM fornisco il giusto slancio per un'attività, portando traffico qualificato sul sito, ma soprattutto, portando in cima ai risultati di ricerca il nome del cliente.

- **Documentazione tecnica.** L'azienda progetta e realizza cataloghi, brochure e qualsiasi documentazione utile a presentare prodotti o attività aziendali, che risultino precisi nei dettagli e graficamente adatti.

- **Eventi.** Claim si occupa della realizzazione grafica dello stand dei propri clienti durante le fiere di settore, soprattutto all'estero, senza sottovalutare la preparazione di materiale cartaceo, fondamentale ad un evento fieristico.

Durante le nove settimane di permanenza in azienda, ho ricoperto varie mansioni su progetti diversi, che mi hanno permesso di acquisire nuove competenze, consolidando e ampliando le conoscenze pregresso. Ad esempio, la configurazione ambiente di lavoro locale (Sublime Text, wamp, google drive per i backup, adobe photoshop, strumenti di debug, phpMyAdmin e download db in locale e la creazione di newsletter con tabelle (html + css + photoshop). L'utilizzo di strumenti di debug per chrome o mozilla (firebug e console per sviluppatori) per verificare la presenza di errori nelle traduzioni; In particolare, mi sono occupato dell'assistenza e della manutenzione ai siti dei clienti. Inoltre ho contribuito allo sviluppo del CRM online Bconsole. Si tratta di un gestionale su cloud che raccoglie tutti i principali strumenti di gestione aziendale. Oltre alla programmazione vera e propria, mi è stata fornita una formazione di base sui linguaggi di programmazione e sui principali applicativi usati dai miei colleghi.

In questa tesi mi focalizzerò sul progetto Bconsole ed, in particolare, sull'incarico principale che mi è stato affidato, ossia interfacciare l'applicazione con le API di Google Charts. Questa necessità deriva dal fatto di avere degli strumenti aggiuntivi di supporto, che ci permettono di prendere le decisioni migliori e in maniera più rapida per il nostro business, come le mappe o i grafici stabili, semplici da usare e sempre aggiornati. Durante il tirocinio ho realizzato un modulo di report con dei grafici.

Questa applicazione usa una pagina HTML come interfaccia grafica (GUI), da dove è possibile, attraverso delle select e dei checkbox, impostare i parametri di ricerca. In questo file, sono importati il file di stile CSS, jQuery una libreria

ovvero comprende tutte le attività atte a generare traffico qualificato verso un determinato sito web

JavaScript, DataTables un plugin di jQuery, lo script jsapi di Google API e lo script JavaScript che esegue la chiamata ad una applicazione Laravel/PHP che interroga il database, disegna ed esegue un controllo degli errori. Per lo sviluppo in locale è stato utilizzato il server built-in di Laravel e Wamp, per la configurazione, importazione e debug del database è stato usato MySQL Workbench e phpMyAdmin, per scrivere il codice invece si è usato PhpStorm. L'applicazione contiene la copia di un database di un cliente prima pulito da dati sensibili e poi ripopolato con dati fake tramite script PHP. Nel Capitolo 2 vedremo framework e modelli utilizzati per realizzare questa applicazione web. Nel capitolo 3 vi sarà una descrizione del progetto in tutte le sue fasi. Infine, nel Capitolo 4 sono stati allegati alcune parti di codice dell'applicazione web.

Capitolo 2

Framework e Modelli Utilizzati

In questo capitolo vengo descritti i modelli, i framework utilizzati in tutto il percorso di elaborazione della tesi. Sotto la sezione "Altro" troviamo, inoltre, degli approfondimenti su due concetti di economia aziendale che ho affrontato durante il tirocino,

2.1 Modelli

2.1.1 Modello Client-Server

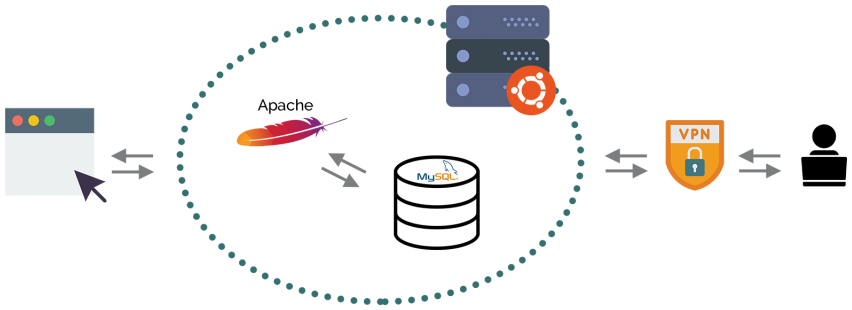

Figura 2.1: Modello Client Server

I calcolatori collegati in rete comunicano seguendo il modello client-server. In questo modello client-server la comunicazione avviene sempre tra una coppia di elementi detti appunto Client e Server.

Con Client si intende di solito l'elemento della coppia che attiva o richiede un servizio di comunicazione

Il Server rappresenta il fornitore del servizio richiesto.

Un esempio di questo modello un browser (web client dell'utente) e un Web Server (nel mio caso Apache) sono la coppia di elementi attiva in una comunicazione Web

2.1.2 Ajax Call

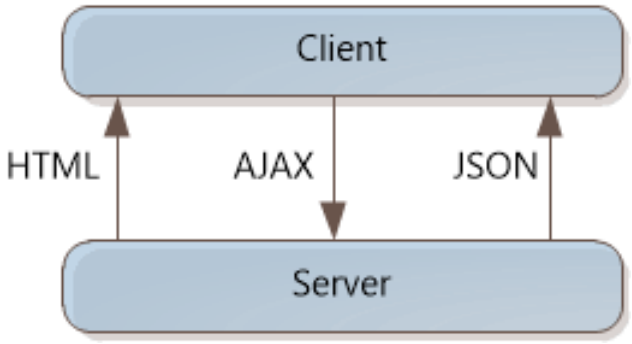

Figura 2.2: Ajax Call

In informatica AJAX, acronimo di Asynchronous JavaScript and XML, è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive (Rich Internet Application).

Nello specifico quando parliamo di chiamata Ajax ci riferiamo ad un oggetto specifico: XMLHttpRequest. Ogni browser ha una sua documentazione sull'implementazione di tale oggetto a volte lo troviamo anche con un nome diverso.

Nel caso di Internet Explorer, ad esempio, questo oggetto è restituito da un ActiveXObject, mentre nei browsers alternativi più diffusi (Mozilla, Safari, FireFox, Netscape, Opera ed altri) XMLHttpRequest è supportato nativamente, cosa che dovrebbe accadere anche per IE dalla versione 7.

Questo oggetto permette di inviare una richiesta in background al server senza ricaricare la pagina dell'utente e senza che lo stesso se ne renda conto. Le richieste si dicono di tipo asincrono, il che significa che non bisogna necessariamente attendere che sia stata ultimata per effettuare altre operazioni, stravolgendo sotto diversi punti di vista il flusso dati tipico di una pagina web. Le richieste che possono essere inviate tramite chiamata ajax sono:

- POST
- GET
- PUT
- DELETE
- PATCH

In questo progetto ho implementato la chiamata ajax usando la funzione ajax() del framework jQuery per inviare una richiesta al server PHP/Laravel.

2.1.3 Pattern MVC

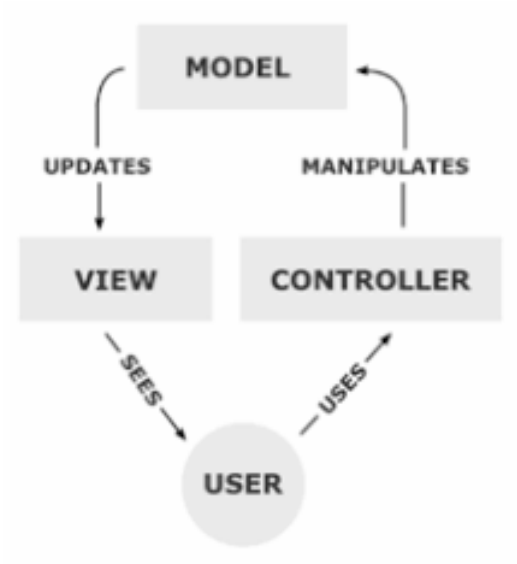

Figura 2.3: Pattern MVC

Il pattern MVC è uno dei più diffusi pattern architetturali in ambito informatico. Il modello semplifica e ordina il codice dell'applicazione aumentando la velocità e la chiarezza nello sviluppo. Il pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali:

- Model: si occupa di dialogare con il database
- Controller: gestisce le richieste dell'utente, modificando lo stato degli altri 2 componenti
- View: permette la visualizzazione dei dati degli altri stati e permette l'interazione tra utenti e agenti

2.2 Framework

2.2.1 Laravel

Laravel è un framework PHP che permette di sviluppare applicazioni dai piccoli progetti fino a quelli a livello enterprise. Il software viene distribuito con licenza MIT dal suo creatore Taylor Otwell che mantiene anche tutto il codice utilizzando una repository su GitHub.

Nel 2013 Laravel diventa il più popolare framework PHP, seguito da Phalcon, Symfony 2, CodeIgniter e altri. Ad agosto 2014, Laravel risulta essere il progetto PHP più seguito su GitHub.

Figura 2.4: Logo Laravel

2.2.2 Google Api Chart

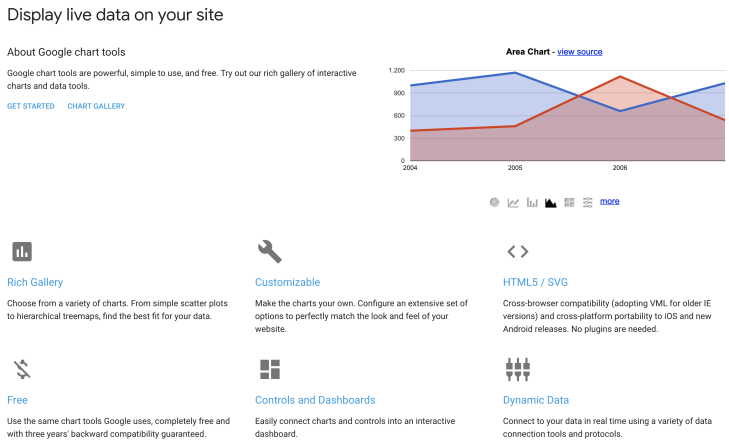

Figura 2.5: Google Chart API

Google Image Charts API è una soluzione molto comoda, semplice ed efficace per creare grafici su un sito web. Il servizio è completamente gratuito e utilizzabile anche su prodotti commerciali, anche se ovviamente ha delle limitazioni d'uso che impediscono alcuni utilizzi impropri, ad esempio non è possibile utilizzare le API al di fuori di siti web o in applicazioni client.

2.2.3 jQuery

Figura 2.6: jQuery

jQuery è una libreria JavaScript per applicazioni web. Nasce con l'obiettivo di semplificare la selezione, la manipolazione, la gestione degli eventi e l'animazione di elementi DOM in pagine HTML, nonché implementare funzionalità AJAX.

Le sue caratteristiche permettono agli sviluppatori JavaScript di astrarre le interazioni a basso livello tra interazione e animazione dei contenuti delle pagine. L'approccio di tipo modulare di jQuery consente la creazione semplificata di applicazioni web e versatili contenuti dinamici.

È un software libero, distribuito sotto i termini della Licenza MIT.[1] Al 2018, jQuery risulta la libreria JavaScript più utilizzata su Internet.

2.2.4 Materialize CSS

Figura 2.7: Materialize CSS

Materialize è un framework js/css basato sui principi dettati da Google per creare un design material. Questo package è molto utile per sviluppare layout frontend. Per design Material si intende un linguaggio di progettazione che combina i principi classici del design di successo insieme a innovazione e tecnologia. L'obiettivo di Google è quello di sviluppare un sistema di progettazione che consenta un'esperienza utente unificata su tutti i loro prodotti su qualsiasi piattaforma.

2.3 Altro

2.3.1 CRM

Figura 2.8: CRM

Acronimo di Customer Relationship Management, un sistema CRM è conosciuto come un modo efficace per gestire i rapporti con i clienti. Il CRM serve alle aziende per individuare e gestire i profili di clienti acquisiti e potenziali, così da mettere a punto attività e strategie che da un lato aiutino a catturare nuovi clienti e dall'altro massimizzare i profitti sui clienti fedeli, cercando di comprenderne esigenze e aspettative. Il CRM può essere scomposto in tre macro aree: operativo, analitico e collaborativo mentre l'architettura tecnologica rappresenta il substrato che ne rende possibile l'applicazione. Il CRM operativo è costituito dalle applicazioni CRM rivolte al cliente che supportano le attività di:

- back office per la gestione degli ordini;
- della supply chain e delle transazioni con il sistema informatico dell'azienda;
- le attività di front office per l'automazione delle forze vendita e l'automazione del marketing d'impresa;
- le attività di mobile office per il supporto alle attività degli agenti e per altri servizi di supporto. Il CRM analitico indica la fase di raccolta e di analisi dei dati, che permette di organizzare la conoscenza a supporto delle decisioni del management. Il CRM collaborativo consente di instaurare rapporti personalizzati con il cliente attraverso i molteplici canali a disposizione. E' costituito da tutti i diversi strumenti di comunicazione con i quali un cliente

potrebbe interagire, ad esempio le e-mail, le chiamate telefoniche, i fax, le pagine web.

2.3.2 ERP

Figura 2.9: ERP

Con l'acronimo ERP si fa riferimento all'Enterprise Resource Planning. Si riferisce a una suite di software che le organizzazioni utilizzano per gestire le attività commerciali quotidiane, come ad esempio contabilità, procurement, project management, gestione del rischio e compliance e operations per la supply chain. Una suite ERP completa include anche enterprise performance management, un software che aiuta a pianificare, prevedere e comunicare i risultati finanziari di un'organizzazione.

I sistemi ERP uniscono e definiscono un insieme di processi di business e ne garantiscono lo scambio di dati. Grazie alla raccolta di dati transazionali condivisi provenienti da diverse fonti dell'organizzazione, i sistemi ERP eliminano la duplicazione dei dati e ne garantiscono l'integrità tramite una "single source of truth".

Oggigiorno, i sistemi ERP sono fondamentali per la gestione di migliaia di aziende di tutte le dimensioni e appartenenti a settori diversi. Per queste aziende, l'ERP è tanto importante quanto l'elettricità che alimenta tutti i sistemi.

Capitolo 3

Descrizione Progetto

In questo capitolo verrà descritto per intero il progetto, partendo dall'analisi dei requisiti, fino alla progettazione e alla messa online dell'applicazione, indicando anche i software utilizzati.

3.1 Analisi dei Requisiti

3.1.1 Fase 1

In una prima fase, durante il tirocinio, mi è stato chiesto di sviluppare un'applicazione web ad alto livello che avesse le seguenti funzionalità:

- Interrogazione del database del gestionale Bconsole
- Visualizzazione dei dati su grafici a torta
- Possibilità di raggruppare i dati per anno
- Possibilità di scegliere la nazionalità dei clienti da includere o escludere dalla ricerca
- Possibilità di visualizzare nel grafico a torta il numero di account, il numero di ordini effettuati e l'importo del fatturato
- Ottimizzazione delle query per avere alte performance, anche in caso di grandi moli di dati
- Utilizzare PHP puro senza l'utilizzo di framework per interrogare il database
- Utilizzare una libreria di appoggio a scelta per la visualizzazione dei grafici
- Creare una interfaccia grafica base

3.1.2 Fase 2

In seguito, dopo aver terminato il tirocinio, ho continuato ad occuparmi di sviluppo software e, quindi, ho deciso di ampliarne i requisiti:

- Installare un framework backend (Laravel) per migliorare la gestione dell'applicazione
- Installare un framework frontend (Materialize CSS) per migliori l'interfaccia grafica e l'interazione utente
- Rendere responsive l'applicazione
- Creare una pagina lista con paginazione e dettaglio per Ordini
- Creare una pagina lista con paginazione e dettaglio per Anagrafica
- Creare una pagina lista e per Paginazione
- Aggiunta di un campo al database geografico per visualizzare le bandiere degli stati
- Creare uno script PHP che popola il db con i dati mancanti (dati volutamente oscurati dopo l'export per questioni di privacy)

3.2 Progettazione Concettuale

3.2.1 Modello Concettuale: Entità-Associazione (E-R)

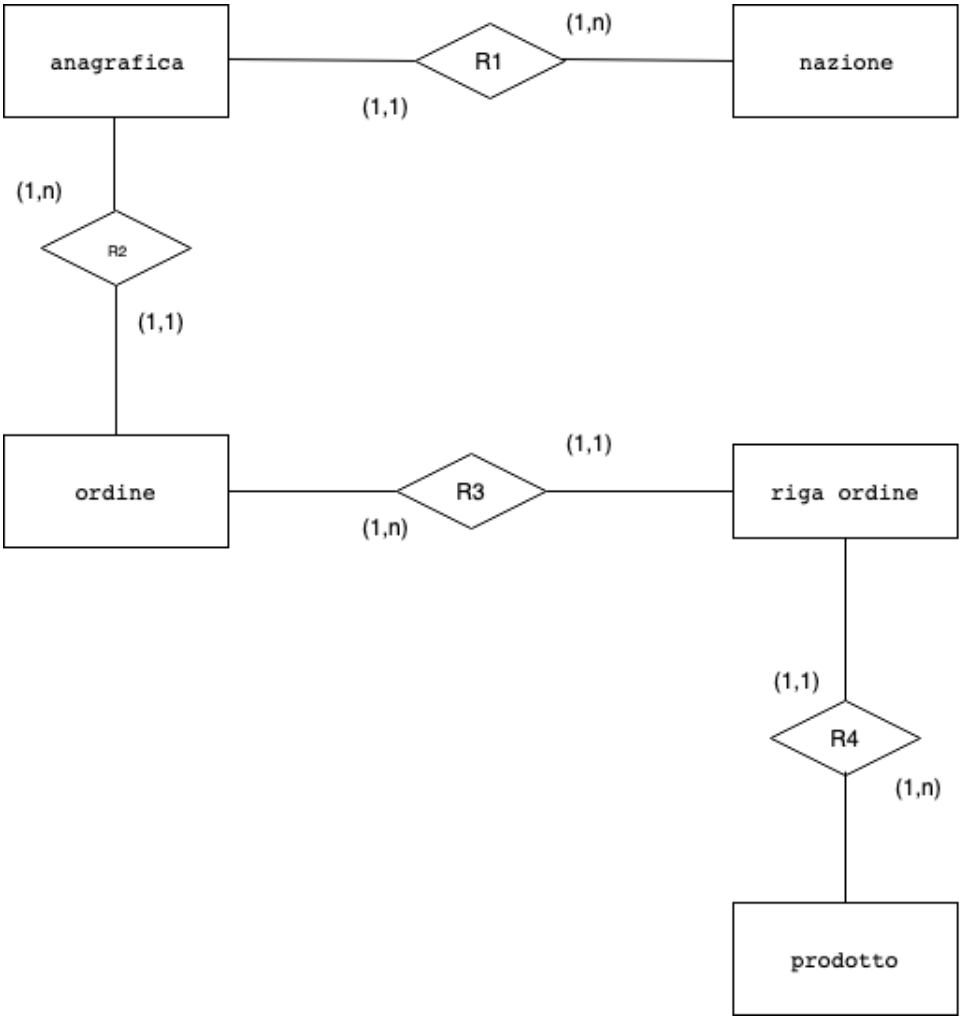

Figura 3.1: Schema ER

3.2.2 Dizionario dei dati

Entità

Entità	Descrizione	Attributi	Identificatore
ana	Anagrafica, lista clienti	id_ana, cod_ana, id_cat, id_age, is_frn, id_tip_pag, id_iva, id_set, id_naz, id_cmn, id_prv, id_reg, anacol, note, note_2, avviso, is_blk, alt, created_at, updated_at, created_by, updated_by, id_cen_cst, fld_custom_1, fld_custom_2, fld_custom_3, fld_custom_4, fld_custom_5, id_org, is_new, fld_custom_6, fld_custom_7, fld_custom_8, fld_custom_9, fld_custom_10, id_lst, id_cen_ric, updated_doc, dsc_iva, id_ext, fld_custom_11, fld_custom_12, fld_custom_13, fld_custom_14, fld_custom_15, fld_custom_16, fld_custom_17, fld_custom_18, fld_custom_19, fld_custom_20, sco, sco_sec, sco_thi, sco_for, cat_cus1, cat_cus12, cat_cus2, cat_cus22, last_mod_ext, is_pcp, perc_pcp, is_prf, perc_prf, ipa, rif_amm, codeori	id_ana

dat_cit	Database Geografico	cit_idd, cit_cod_cit, cit_cap, cit_nom, cit_cod_prv, cit_nom_prv, cit_reg, citt_pre_tel, cit_not, cit_cor, iso, name, alpha_code_iso	cit_idd
cfn	Ordine	id_cfn, id_doc_prt, num_cfn, num_cfn_ext, dat_cfn, id_ana, id_sta, id_tip_pag, id_alg, dsc, rif, note, created_at, updated_at, created_by, updated_by, id_ban_ana, id_ban, dat_cns, note_alert, note_int, note_cns, is_sent, id_ext, id_ana_fin	id_cfn
cfn_rig	Riga dell'ordine	id_cfn_rig, id_cfn, id_pro, id_sta, dsc, dsc_lng, dsc_agg, rif, prz_uni, id_iva, qta, imp, sco, sco_sec, sco_thi, sco_for, dat_cns, id_ums, created_at, updated_at, created_by, updated_by, ord_vis, tot_imp, id_cen_ric, id_mag, id_rig_prt, tipo_doc_prt	id_cfn_rig

pro	Listino Prodotti	id_pro, cod_pro, dsc, id_cat, id_sub_cat, id_sub_cat3, dsc_lng_2, dsc_lng, prz_lst, prz_acq, pes_uni, dim_lar, dim_alt, dim_prf, pth_img, note, created_at, updated_at, created_by, updated_by, obb, peso_imb, dim_imb_lar, dim_imb_alt, dim_imb_prf, id_sta, id_tip, id_cen_cst, id_cen_ric, id_ums, id_iva, qta_imb, is_var, is_not_sta, is_web, is_web_prt, iso, fld_custom_1, fld_custom_2, fld_custom_3, fld_custom_4, fld_custom_5, fld_custom_6, fld_custom_7, fld_custom_8, fld_custom_9, fld_custom_10, fld_custom_11, fld_custom_12, fld_custom_13, fld_custom_14, fld_custom_15, fld_custom_16, fld_custom_17, fld_custom_18, ord_wis_web, dsc_lng_3, dsc_lng_4, ord_vis_web, is_kit, ric_prz_acq, sco_prz_lst, prz_kit, id_ana, not_in_prv, is_opt, id_pro_ext, del_ext, last_mod_ext, to_upd, is_new, is_upd	id_pro

Associazioni

Associazione	Attributi	Entità Collegate
R1		ana (1,1), dat_cit(1,n)
R2		ana (1,n), cfn(1,1)
R3		cfn (1,n), cfn_rig(1,1)
R4		cfn_rig (1,1), pro(1,n)

3.3 Software Utilizzati

3.3.1 PhpStorm

Figura 3.2: Php Storm

E' uno tra i più giovani degli IDE, ma è di certo uno dei migliori del mercato come traspare dal loro payoff "Lightning-smart PHP IDE". JetBrains, l'azienda proprietaria del software, parte con l'idea di lanciare un IDE Java con supporto a Java e Android. Poi, inizia a rilasciare il supporto e altre release per i seguenti linguaggi: HTML, JavaScript, CSS, Sass, LESS, TypeScript, CoffeeScript, Node.js, PHP, Ruby on Rails e Python/Django.

Le sue caratteristiche principali sono:

- Autocompletamento del codice
- PHP Refactoring
- Suggerimenti per l'inserimento dei parametri
- Code (Re)arranger
- Code Quality Analysis
- Semplice navigazione e ricerca nel codice
- Debugging, Testing and Profiling
- Supporto per i principali Frameworks PHP

3.3.2 MySQL Workbench

Figura 3.3: Mysql Workbench

MySQL Workbench è un tool creato per database architects, developers, and DBAs. MySQL Workbench è disponibile per Windows, Linux and Mac OS X.

Questo programma è molto utile per:

- Progettazione Database
- Sviluppo in linguaggio MySql
- Amministrare il Database
- Monitorare le performance del Database tramite una dashboard grafica
- Database Migration - Import e Backup veloce di Database

3.3.3 Google Chrome

Figura 3.4: Google Chrome

È un browser web tra i più utilizzati, basato su Blink e sviluppato da Google. Di seguito alcune statistiche di utilizzo nei browser nel mondo.

Come possiamo vedere Chrome è uno dei browser più utilizzati al mondo, motivo per il quale quando si sviluppa una applicazione web viene scelto come browser di test. In aggiunta a questo, Chrome può vantare di una sezione riservata agli sviluppatori per avere informazioni aggiuntive sulla pagina che si sta guardando e debuggare Javascript in tempo reale. Se tutto ciò non è sufficiente per il progetto in cui si sta lavorando, è sempre possibile installare estensioni di terze parti per estendere queste funzionalità.

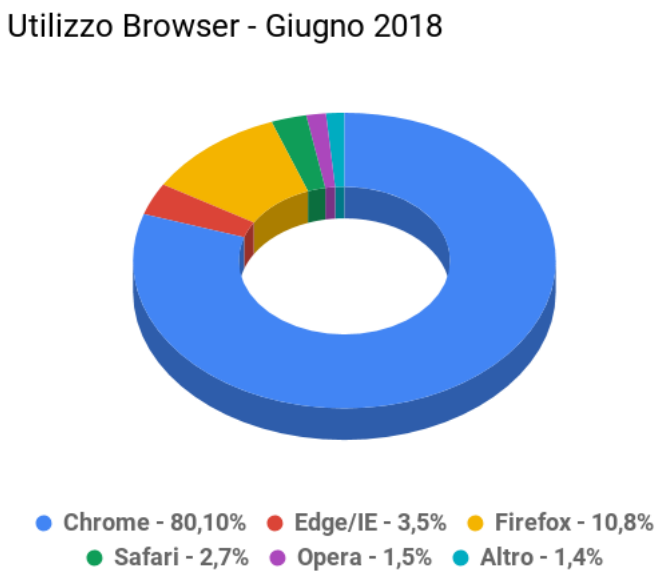

Figura 3.5: Browsers Chart

Figura 3.6: Bitbucket

3.3.4 Bitbucket

Bitbucket è un servizio di hosting web-based per progetti che usano Git o Mercurial. Bitbucket offre sia piani commerciali che account gratuiti. Esso offre account gratuiti ed un numero illimitato di repository privati dal settembre 2010. Bitbucket è scritto in Python usando il Framework per applicazioni web Django

Bitbucket era precedentemente una startup indipendente, fondata da Jesper Nøhr. Il 29 settembre 2010, Bitbucket fu acquisita da Atlassian. Inizialmente Bitbucket offriva supporto di hosting solo per progetti Mercurial. Il 3 ottobre 2011 Bitbucket annunciò ufficialmente il supporto per hosting Git

3.3.5 Sourcetree

SourceTree è un applicativo desktop che permette di usare git tramite interfaccia grafica e di gestire le repository presenti su Bitbucket e in locale.

Figura 3.7: SourceTree

Figura 3.8: Git

3.3.6 Git

Git è un software di controllo versione distribuito utilizzabile da interfaccia a riga di comando, creato da Linus Torvalds nel 2005. Git immagazzina e tratta le informazioni in modo molto diverso dagli altri sistemi, anche se l'interfaccia utente è abbastanza simile; La principale differenza tra Git e gli altri VCS (incluso Subversion), è come Git considera i suoi dati. Concettualmente la maggior parte degli altri sistemi salvano l'informazione come una lista di modifiche ai file. Questi sistemi (CVS, Subversion, Perforce, Bazaar), considerano le informazioni che mantengono come un insieme di file, con le relative modifiche fatte ai file nel tempo. Git non considera i dati né li registra in questo modo. Infatti, Git considera i propri dati come una serie di istantanee (snapshot) di un mini filesystem. Ogni volta che committi, o salvi lo stato del tuo progetto in Git, fondamentalmente il software fa un'immagine di tutti i file in quel momento, salvando un riferimento allo snapshot. Per essere efficiente, se alcuni file non sono cambiati, Git non li risalva, ma crea semplicemente un collegamento al file precedente già salvato. Questa è una distinzione importante tra Git e pressocché tutti gli altri VCS. Git riconsidera quasi tutti gli aspetti del controllo di versione che la maggior parte degli altri sistemi ha copiato dalle generazioni precedenti. Questo rende Git più simile a un mini filesystem con a disposizione strumenti incredibilmente potenti che un semplice VCS. La maggior parte delle operazioni in Git, necessitano solo di file e risorse locali per operare — generalmente non occorrono informazioni da altri computer della rete. Git, quindi, non ha bisogno di connettersi al server

per scaricare un progetto, ad esempio, e per poi mostrarlo: lo legge direttamente dal database locale. Questo significa è possibile vedere il progetto quasi istantaneamente. Per verificare le modifiche introdotte tra la versione corrente e la versione meno recente di un file, Git può accedere a quest'ultimo file e calcolare le differenze localmente, invece di dover chiedere a un server remoto di farlo o di scaricare dal server remoto una versione precedente del file, per poi farlo in locale.

in questo modo, è possibile lavorare offline, ovvero senza connessione alla VPN. In molti altri sistemi questo è impossibile o molto complesso.

Con Perforce, per esempio, sono molto limitate le funzioni senza connessione al server; con Subversion e CVS, invece, è possibile modificare i file, ma non inviare le modifiche al proprio database, perché il database è offline. Qualsiasi cosa in Git è controllata, tramite checksum, prima di essere salvata ed è referenziata da un checksum. In questo modo è impossibile cambiare il contenuto di qualsiasi file o directory senza che Git lo sappia. Questa è una funzionalità interna, di basso livello di Git, ed è intrinseco della sua filosofia. Non può succedere che alcune informazioni in transito si possano perdere o che un file si corrompa senza che Git non sia in grado di accorgersene. Il meccanismo che Git usa per fare questo checksum è un hash chiamato SHA-1. Si tratta di una stringa di 40-caratteri, composta da caratteri esadecimali (0–9 ed a–f) e calcolata in base al contenuto di file o della struttura della directory in Git. Infatti, Git salva qualsiasi cosa nel suo database, e il riferimento ad un file non è basato sul nome del file, ma sull'hash del suo contenuto. Infine, i file in Git possono assumere "tre stati": committed (committati), significa che il file è al sicuro nel database locale modified (modificati), significa che il file è stato modificato, ma non è ancora stato committato nel database. staged (in stage), modificato in stage significa che un file è stato contrassegnato, modificato nella versione corrente, perché venga inserito nello snapshot alla prossima commit. Questo ci porta alle tre sezioni principali di un progetto Git: la directory di Git, la directory di lavoro e l'area di stage.

3.4 Implementazione Progetto

3.4.1 PHP Version

In questa versione dell'applicazione non ho utilizzato nessun framework CSS. Mi sono concentrato, infatti, sul funzionamento dell'applicazione e sulle sue perfomance. Quindi, dopo aver scaricato jQuery e Google Chart API, ho inziato a scrivere lo script che lanciava la richiesta al server. In un secondo momento ho poi sviluppato la query sul database esistente di Bconsole. In seguito, per effettuare i primi test, ho dovuto importare un database di un cliente reale che abbia effettuato numerosi ordini all'estero negli ultimi anni, così da poter avere molti dati da visualizzare nel grafico a torta. Una volta esportato il database, ho dovuto procedere alla cancellazione dei dati dalle anagrafiche per motivi di privacy. Per quanto riguarda il backend, invece, ho esposto i seguenti servizi:

- Lista nazioni: Tale servizio restituisce la lista nazioni per popolare la select a frontend.
- Ricerca: Questo servizio riceve i parametri di ricerca in input e fornisce i dati trovati da passare allo script js per generare il grafico a torta tramite le API di Google Chart

3.4.2 Laravel Version

In un secondo momento, dopo aver concluso il tirocinio, ho svolto altre esperienze lavorative che mi hanno permesso di ampliare le mie conoscenze in ambito di sviluppo software. Ho deciso quindi, di revisonare l'applicazione. Per renderla più completa, ho aggiunto delle schermate di Lista (con ricerca rapida) e Dettaglio. In seguito ho convertito il progetto da PHP base in Laravel, un framework web che semplifica la gestione di questo tipo di applicazioni. Inoltre, dal punto di vista grafico, ho installato Materialize CSS per migliorare la grafica e semplificare l'interazione all'utente. Per poter visualizzare correttamente la pagina degli Ordini e delle Anagrafiche ho popolato le ragioni sociali utilizzando una libreria chiamata Faker di fzaninotto [1]. Infine, nella pagina delle nazioni aggiungendo e popolando un campo chiamato alpha_code_iso sono riuscito a visualizzare le bandiere associate.

[1] Laravel package Faker - https://github.com/fzaninotto/Faker

Capitolo 4

Codice Progetto

In questo capitolo vengono presentate le parti principali dell'appicazione sviluppata, oppurtanamente divisi per linguaggio (*Javascript*, *PHP + HTML*, *SQL*) e per versione dell'applicazione. Il codice *Javascript* e il codice *SQL* sono presenti in una sola versione, perchè lo stesso è il medesimo per le due versioni dell'applicazione sviluppata.

4.1 Report JS

4.1.1 Ajax Request

Codice 4.1: Ajax Request

```
var id_plot_account = "account";
var id_plot_imponibile = "imponibile";
var id_plot_ordini = "ordini";
var data_account = "";
var data_imponibile = "";
var data_ordini = "";
var current_chart_visibile = "";


function show_loading(cssclass) {

    $('.' + cssclass).show();
}

function hide_loading(cssclass) {

    $('.' + cssclass).hide();

}

function hide_result() {

    $('#result').empty();

}

function draw_table(obj) {

    var id_table = "report-table";
```

```
        var str = "<table id='" + id_table + "' class='hoverable responsive-table'
cellspacing='0' width='100%'>" +
            "<thead>" +
            "<tr>" +
            "<th>Paese</th>" +
            "<th>Account</th>" +
            "<th>Iimponibile Totale</th>" +
            "<th># Ordini</th>" +
            "</tr>" +
            "</thead>" +
            "<tbody>";

        $.each(obj, function (key, value) {
            str += "<tr>";
            str += "<td>" + value.paese + "</td>;
            str += "<td>" + value.account + "</td>";
            str += "<td>" + format_euro(value.tot) + "</td>";
            str += "<td>" + value.n_ordini + "</td>";
            str += "</tr>";
        });

        str += "</tbody>" +
            "</table>";

        $('#table').html(str);
        id_table = "#" + id_table;

        return id_table;
    }


    function resize() {

        var colonna = $('#colonna').val();
        $("#plot div[name='plot']").show();
        draw_plot(data_account, id_plot_account, id_plot_account);
        draw_plot(data_imponibile, id_plot_imponibile, id_plot_imponibile);
        draw_plot(data_ordini, id_plot_ordini, id_plot_ordini);
        show_plot_select('sel', colonna);
    }
    $(window).resize(function(){
        resize();
    });


    function draw_plot(data, id_plot, titolo) {
        var options = {
            title: titolo,
            is3D: 'true',
            width: '100%',
            height: '100%',
            legend: {position: 'bottom', alignment: 'start'},
        };

        var chart = new google.visualization.PieChart(document.getElementById(id_plot));
        chart.draw(data, options);
    }

    function show_plot_select(id_select, id_plot) {
        $("#plot div[name='plot']").hide();
        $('#' + id_plot).show();
        $('#' + id_select).show();

    }

    function jsontogooglejson(obj, id_plot) {
        //creo l'array paese
        var paese = [];
```

```
        var data = new google.visualization.DataTable();
        $.each(obj, function (key, value) {
            paese.push(value.paese);
        });

        if (id_plot === "account") {
            //creo l'array account
            var account = [];
            $.each(obj, function (key, value) {
                account.push(parseInt(value.account));
            });

            data.addColumn('string', 'paese');
            data.addColumn('number', 'account');

            for (i = 0; i < account.length; i++)
                data.addRow([paese[i], account[i]]);
        }
        else if (id_plot === "imponibile") {
            //creo l'array imponibile
            var tot = [];
            var val = 0;
            $.each(obj, function (key, value) {
                val = parseInt(value.tot);
                tot.push(val);
            });
            data.addColumn('string', 'paese');
            data.addColumn('number', 'imponibile');

            console.log(tot);
            for (i = 0; i < tot.length; i++)
                data.addRow([paese[i], tot[i]]);

        }
        else if (id_plot === "ordini") {
            //creo l'array ordini
            var ordini = [];
            $.each(obj, function (key, value) {
                ordini.push(parseInt(value.n_ordini));
            });

            data.addColumn('string', 'paese');
            data.addColumn('number', 'ordini');

            for (i = 0; i < ordini.length; i++)
                data.addRow([paese[i], ordini[i]]);
        }
        return data;
    }

    function get_data_post(id_form) {
        var type = "type=";
        var anno = "anno=" + $("#anno").val();

        var data_post = "";
        var paesi = "paesi=";
        var count_all = $("input:checkbox").length;
        var count_check = $("input:checkbox:checked").length;
        var count = Math.round(count_all / 2);

        if (count >= count_check || count_all == count_check) {//transmit only the checked
            type += "1";

            $("input:checkbox:checked").each(function () {
                paesi += $(this).attr("value") + ";";
            });
        }
        else {//transmit only the NOT checked
```

```
        type += "0";

        $("input:checkbox:not(:checked)").each(function () {
            paesi += $(this).attr("value") + ";";
        });

    }


    data_post = type + "&" + paesi + "&" + anno;
    return data_post;
}

//Format value in Euro for it-IT locale - Ex. 2533.13 -> 2.533,13 euro
function format_euro(value){
    return Number(value).toLocaleString("it-IT", {minimumFractionDigits: 2}) + " euro";
}

$("#form_search").submit(function (e) {
    e.preventDefault(); //STOP default action

    var c = $("#check input:checkbox").length == 0;
    var cc = $("#check input:checkbox:checked").length == 0;
    if (c) {
        $('#error').html('Scegli un\'altro anno, questo non ha nemmeno una nazione')
        .fadeIn('slow')
        .delay(4000)
        .fadeOut('slow');
    }
    else if (cc) {
        $('#error').html('Seleziona almeno una nazione')
        .fadeIn('slow')
        .delay(4000)
        .fadeOut('slow');
    }
    else {
        $("#table,#account,#imponibile,#ordini").html('');
        $('#sel').hide();
        $('#colonna').val('imponibile');

        var count = $("#check input:checkbox").length;


        var post_data = get_data_post(this);
        //alert(post_data);


        $.ajax({
            type: "POST",
            url: "report/get-chart",
            data: post_data,
            headers: {
                'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
            },
            beforeSend: function () {
                show_loading('loader_chart');
                show_loading('loader_table');

            },
            success: function (data) {
                var obj = data[0];

                var id_table = draw_table(obj);//#example
                hide_loading('loader_table');

                //account
                var data_account = jsontogooglejson(obj, id_plot_account);
                draw_plot(data_account, id_plot_account, id_plot_account);
```

```
                    //imponibile
                    var data_imponibile = jsontogooglejson(obj, id_plot_imponibile);
                    console.log(data_imponibile);
                    draw_plot(data_imponibile, id_plot_imponibile, id_plot_imponibile);

                    // ordini
                    var data_ordini = jsontogooglejson(obj, id_plot_ordini);
                    draw_plot(data_ordini, id_plot_ordini, id_plot_ordini);

                    //mostra imponibile per primo
                    hide_loading('loader_chart');
                    show_plot_select('sel', id_plot_imponibile);


                }
            });
        }


    });

    $('#colonna').change(function (event) {
        event.preventDefault();

        var colonna = $('#colonna').val();
        if (current_chart_visibile != colonna) {

            $("#plot div[name='plot']").hide("slow");
            $('#' + colonna).show("slow");
            current_chart_visibile = colonna;
        }

    });


    $('#anno').change(function (event) {
        $('#action').hide();
        $("#table,#account,#imponibile,#ordini").html('');
        $('#sel').hide();
        $('#colonna').val('imponibile');
        var anno = $('#anno').val();

        if(anno !== ""){
            $('#cerca').show();
            $('#seleziona').show();
            $('#deseleziona').show();
            $('#title-checkbox-list').show();
        }else{
            $('#cerca').hide();
            $('#seleziona').hide();
            $('#deseleziona').hide();
            $('#title-checkbox-list').hide();
        }

        event.preventDefault();
        var anno = $('#anno').val();
        $.ajax({
            type: "GET",
            url: "report/get-nation-by-year/" + anno,
            beforeSend: function () {

                show_loading('loader_check');

            },
            success: function (response) {

                hide_loading('loader_check');
```

```
                var str = "";
                var result = response[0];
                if (result.length === 0)
                    str = "Nessuna Nazione da visualizzare per l'anno " + anno;
                else {
                    $.each(result, function (key, value) {
                        str += "<label for='" + value.id_naz + "'>";
                        str += "<input  type='checkbox' name='paese[]'";
                        str += "class='checkbox-nation' id='" + value.id_naz + "'";
                        str += "value='" + value.id_naz + "'/>";
                        str += "<span>";
                        str += value.cit_nom;
                        str += "</span>";
                        str += "</label>";
                    });
                }
                $("#check").html(str);
            }
        });
    });
```

4.2 Report - PHP version

4.2.1 connect.php

Codice 4.2: connect.php

```
<?php
    GLOBAL $dbh;
    $db_name = "****";
    $host = "localhost";
    $db_user = "****";
    $db_psw = "****";
    $col = "mysql:host=$host;dbname=$db_name";

    try {//tentativo di connessione
        $dbh = new PDO($col , "$db_user", "$db_psw");
    }
    catch(PDOException $e) {//gestione errori
        echo 'Attenzione errore: '.$e->getMessage();
    }
?>
```

4.2.2 elabora.php

Codice 4.3: elabora.php

```
<?php
include ("connect.php");

function get_cond_paese($paesi,$tipo){

    $cond_paese = "";
    if ($tipo == "1"){

        foreach($paesi as $selected){

            $cond_paese .= "id_naz = '".$selected."' OR ";

        }

        $cond_paese = substr($cond_paese, 0, -4);
    }
```

```
    else if ($tipo == "0"){
        foreach($paesi as $selected){

            $cond_paese .= "id_naz <> '".$selected."' AND ";

        }

        $cond_paese = substr($cond_paese, 0, -4);
    }
    else
        $cond_paese = " 0";
    return $cond_paese;
}

function get_cond_anno($anno){
    if ($anno == "tutti")
        $cond_anno = " 1";
    else
        $cond_anno = "dat_cfn BETWEEN '".$anno."-01-01' AND '".$anno."-12-31'";
    return $cond_anno;
}

//elaboro la richiesta
$_POST['paesi'] = substr( $_POST['paesi'], 0,-1);

$paesi = explode(";", $_POST['paesi']);
$tipo = $_POST['type'];
$anno = $_POST['anno'];

$cond_anno = get_cond_anno($anno);
$cond_paese = get_cond_paese($paesi,$tipo);

$sql = "SELECT paese, account, round((sum_imp),2) as sum_imp, n_ordini FROM

(SELECT half.cit_nom AS paese, sum_imp, n_ordini, cit_idd
FROM (SELECT cit_idd, cit_nom, SUM(imp)as sum_imp
FROM (SELECT cit_nom, id_cfn ,cit_idd
FROM (SELECT id_ana, cit_nom,cit_idd
FROM (SELECT id_ana, id_naz
FROM ana WHERE ".$cond_paese.")as an
INNER JOIN (SELECT cit_idd, cit_nom
FROM dat_cit) as nome_cit ON id_naz = cit_idd) as ana_ok
INNER JOIN (SELECT id_cfn,id_ana
FROM cfn WHERE ".$cond_anno.")as cfn_o ON ana_ok.id_ana = cfn_o.id_ana) as cfn_ok
INNER JOIN cfn_rig ON cfn_ok.id_cfn = cfn_rig.id_cfn GROUP BY cit_nom)as half
INNER JOIN (SELECT cit_nom, COUNT(*) as n_ordini
FROM (SELECT id_ana, cit_nom
FROM (SELECT id_ana, id_naz FROM ana WHERE ".$cond_paese.")as an
INNER JOIN
(SELECT cit_idd, cit_nom FROM dat_cit) as nome_cit
ON id_naz = cit_idd) as ana_ok
INNER JOIN (SELECT id_cfn,id_ana FROM cfn WHERE ".$cond_anno.")as cfn_oi
ON ana_ok.id_ana = cfn_oi.id_ana GROUP BY cit_nom)as other_half
ON half.cit_nom = other_half.cit_nom) as one
INNER JOIN (SELECT cit_nom, id_naz, COUNT(*) as account
FROM (SELECT cit_nom, id_naz, d.id_ana
FROM (SELECT cit_nom, id_naz, cfn_ot.id_ana
FROM (SELECT id_naz, cit_nom, id_ana
FROM (SELECT id_ana, id_naz FROM ana WHERE ".$cond_paese.")as an
INNER JOIN dat_cit ON id_naz = cit_idd)as f
INNER JOIN
(SELECT id_cfn,id_ana FROM cfn WHERE ".$cond_anno.") as cfn_ot
ON f.id_ana = cfn_ot.id_ana)as d
GROUP BY id_ana) as r GROUP BY id_naz) as other_one
ON cit_idd = id_naz ORDER BY cit_nom;";


$stmt = $dbh->query($sql);
```

```
$result = $stmt->fetchAll(PDO::FETCH_ASSOC);

$dbh = null;

echo json_encode($result);


?>
```

4.3 Report - Laravel Version

4.3.1 Risultati ricerca e Grafico

Codice 4.4: Risultati ricerca e Grafico

```
Route::post('/report/get-chart', 'ReportController@getChart')->name('report.get-chart');

/**
* @param Request $request
*
* @return \Illuminate\Http\JsonResponse
*/
public function getChart( Request $request ) {
  //Handle request
  $paesi = $request->input('paesi');
  $paesi = substr( $paesi, 0, - 1 );

  $paesi = explode( ";", $paesi );
  $tipo  = $request->input('type');
  $anno  = $request->input('anno');

  $cond_anno  = get_cond_anno( $anno );
  $cond_paese = get_cond_paese( $paesi, $tipo );


  $sql = "
        SELECT paese, account, SUM(round((sum_imp),2) ) as tot, n_ordini FROM
        (SELECT half.cit_nom AS paese, sum_imp, n_ordini, cit_idd
        FROM (SELECT cit_idd, cit_nom, SUM(imp)as sum_imp
        FROM (SELECT cit_nom, id_cfn ,cit_idd
        FROM (SELECT id_ana, cit_nom,cit_idd
        FROM (SELECT id_ana, id_naz
        FROM ana WHERE " . $cond_paese . ")as an
        INNER JOIN (SELECT cit_idd, cit_nom FROM dat_cit) as nome_cit
        ON id_naz = cit_idd) as ana_ok
        INNER JOIN (SELECT id_cfn,id_ana
        FROM cfn WHERE " . $cond_anno . ")as cfn_o
        ON ana_ok.id_ana = cfn_o.id_ana) as cfn_ok
        INNER JOIN cfn_rig ON cfn_ok.id_cfn = cfn_rig.id_cfn
        GROUP BY cit_nom,imp)as half
        INNER JOIN (SELECT cit_nom, COUNT(*) as n_ordini
        FROM (SELECT id_ana, cit_nom
        FROM (SELECT id_ana, id_naz
        FROM ana WHERE " . $cond_paese . ")as an
        INNER JOIN (SELECT cit_idd, cit_nom
        FROM dat_cit) as nome_cit ON id_naz = cit_idd) as ana_ok
        INNER JOIN (SELECT id_cfn,id_ana
        FROM cfn WHERE " . $cond_anno . ")as cfn_oi
        ON ana_ok.id_ana = cfn_oi.id_ana
        GROUP BY cit_nom)as other_half
        ON half.cit_nom = other_half.cit_nom) as one
        INNER JOIN
        (SELECT cit_nom, id_naz, COUNT(*) as account
        FROM (SELECT cit_nom, id_naz, d.id_ana
        FROM (SELECT cit_nom, id_naz, cfn_ot.id_ana
        FROM (SELECT id_naz, cit_nom, id_ana
```

```
            FROM (SELECT id_ana, id_naz
            FROM ana WHERE " . $cond_paese . ")as an
            INNER JOIN dat_cit ON id_naz = cit_idd)as f
            INNER JOIN (SELECT id_cfn,id_ana
            FROM cfn WHERE " . $cond_anno . ") as cfn_ot
            ON f.id_ana = cfn_ot.id_ana)as d
            GROUP BY id_ana) as r
            GROUP BY id_naz) as other_one
            ON cit_idd = id_naz
            GROUP BY paese ORDER BY cit_nom;
            ";

        $result = DB::select( $sql );

        return response()->json( [
            $result
        ] );

    }
```

4.3.2 Lista nazioni

Codice 4.5: Lista nazioni

```
Route::get('/report/get-nation-by-year/{year}','ReportController@getNationByYear')
->name('report.nation-by-year');

/**
 * @param $year
 *
 * @return \Illuminate\Http\JsonResponse
 */
public function getNationByYear( $year ) {
    if ( $year == "tutti" ) {
        $cond_anno = "1";
    } else {
        $cond_anno = "dat_cfn BETWEEN '" . $year . "-01-01' AND '" . $year . "-12-31'";
    }

    $sql = "SELECT id_naz, cit_nom
    FROM (SELECT id_naz
    FROM (SELECT id_naz, id_ana FROM ana) as a
    INNER JOIN (SELECT id_ana
    FROM cfn WHERE " . $cond_anno . " ) as cfn_anno ON a.id_ana = cfn_anno.id_ana
    GROUP BY id_naz)as k
    INNER JOIN dat_cit ON id_naz = cit_idd ORDER BY cit_nom;";

    $result = DB::select( $sql );

    return response()->json( [
        $result
    ] );

}
```

4.3.3 Render Pagina Report

Codice 4.6: Render Pagina Report

```
Route::post('/report/get-chart', 'ReportController@getChart')
->name('report.get-chart');

/**
     * Display the resource.
     *
```

```
     * @return \Illuminate\Http\Response
     */
    public function index() {
            return view( 'report' );
    }

    @extends('layout.default')
@section('content')
    <div class="container">
        <div class="card-panel row">

            <form class="col l12 s12" id="form_search" method="post">

                <div class="input-field col l12 s12">
                    <select name="anno" id="anno">
                        <option value="" disabled selected>Scegli un anno</option>
                        <option value="tutti">tutti</option>
                        <option value="2011">2011</option>
                        <option value="2012">2012</option>
                        <option value="2013">2013</option>
                        <option value="2014">2014</option>
                        <option value="2015">2015</option>
                    </select>
                    <label>Anno:</label>
                </div>
                <div>
                    <h4 id="title-checkbox-list">
                        Lista Nazioni da visualizzare:
                    </h4>
                    @include('include.loader')
                    <div id="check"></div>
                    <span id="error"></span>
                </div>
                <button class="btn waves-effect waves-light" id="cerca"
                type="submit" name="action">
                    <i class="material-icons">
                        search
                    </i>
                    <span class="button-label">Cerca</span>
                </button>
                <button class="btn waves-effect waves-light" id="seleziona">
                    <i class="material-icons">
                        done_all
                    </i>
                    <span class="button-label">Seleziona Tutto</span>
                </button>
                <button class="btn waves-effect waves-light" id="deseleziona">
                    <i class="material-icons">
                        refresh
                    </i>
                    <span class="button-label">Deseleziona Tutto</span>
                </button>
            </form>

            <div id="result">
                @include('include.loader')
                <div class="col s12" id="table"></div>
                <div class="row">
                    <div class="col s12" id="plot">
                        <div class="container" name="plot" id="account"></div>
                    </div>
                    <div class="col s12" id="plot">
                        <div class="container" name="plot" id="imponibile"></div>
                    </div>
                    <div class="col s12" id="plot">
                        <div class="container" name="plot" id="ordini"></div>
                    </div>
                    @include('include.loader')
```

```
                    </div>
                    <div id="sel" class="input-field col s12" style="display: none;">
                        <select name="colonna" id="colonna">
                            <option value="" disabled>Scegli un grafico</option>
                            <option value="account">account</option>
                            <option value="imponibile" selected>imponibile</option>
                            <option value="ordini">ordini</option>
                            <label>Colonna:</label>
                        </select>
                    </div>

                </div>
            </div>
        </div>
    @stop
```

4.4 Pagine Lista Dettaglio - Laravel Version

4.4.1 Homepage

Codice 4.7: Homepage

```
//Route
Route::get('/', 'HomeController@index')->name('homepage');
Route::get('/home', function () {
  return redirect()->route('Home');
});

Route::get('/homepage', function () {
  return redirect()->route('Home');
});

//HomeController Method
public function index()
{
 return view('home');
}

@extends('layout.default')
@section('content')
    <div class="container">
        <div class="row">
            <div class="col s12 l12">
                <div class="card-panel row">
                    <h3>Tesi</h3>
                    <span>di Giacomo Fabbian</span>
                </div>
            </div>
        </div>
    </div>
@stop
```

4.4.2 Lista Ordini

Codice 4.8: Lista Ordini

```
Route::get('/orders','OrderController@index')->name('order.index');


    /**
         * Display a listing of the resource.
         *
         * @return \Illuminate\Http\Response
```

```
 */
public function index()
{
    $orders = new Order;
    $orders = $orders->all();

    return view('order.index',compact('orders'));

}

@extends('layout.default')
@section('content')
    <div id = "container" class="container">
        <div class="row">
            <div class="col s12 l12">
                <div class="card-panel row">
                    <table class="responsive-table">
                        <thead>
                        <tr>
                            <th>Numero</th>
                            <th>Data</th>
                            <th>Ragione Sociale</th>
                            <th>Totale</th>
                        </tr>
                        </thead>
                        <tbody>
                        @foreach($orders as $order)
                            <tr>
                                <td>
                                <a href="{{ route('order.show', [$order->id_cfn]) }}">
                                    #{{ $order->id_cfn }}
                                </a>
                                </td>
                                <td>
                                    {{ format_it_date($order->dat_cfn) }}
                                </td>
                                <td>
                                    <a href="{{
                                    route('customer.show',
                                     [$order->customer->id_ana]) }}">
                                        <span class="
                                        flag-icon flag-icon-{{
                                        strtolower($order->customer->nation->alpha_code_iso)
                                        }}
                                         flag-icon-squared"></span>
                                        {{ $order->customer->fld_custom_1 }}
                                    </a>
                                </td>
                                <td>{{ $order->totalOrder() }}</td>
                            </tr>
                        @endforeach
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
@stop
```

4.4.3 Dettaglio Ordine

Codice 4.9: Dettaglio Ordine

```
Route::get('/orders/{id}','OrderController@show')->name('order.show');
```

```
/**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
 */
public function show($id)
{
    $order = new Order;
    $order = $order->find($id);

    return view('order.show',compact('order'));
}

@extends('layout.default')
@section('content')
    <div id = "container" class="container">
        <div class="row">
            <div class="col l12">
                <div class="card-panel row">
                    <h2>Ordine #{{ $order->id_cfn }}</h2>
                    Cliente: <a href="{{ route(
                    'customer.show',
                    [$order->customer->id_ana]) }}">
                        <span class="
                        flag-icon flag-icon-{{
                        strtolower($order->customer->nation->alpha_code_iso) }}
                        flag-icon-squared"></span>
                        {{ $order->customer->fld_custom_1 }}
                    </a>
                    Data: {{ format_it_date($order->dat_cfn) }}
                    <table class="responsive-table">
                        <thead>
                        <tr>
                            <th>Numero</th>
                            <th>Descrizione</th>
                            <th>QuantitÃă</th>
                            <th>Imbonibile</th>
                            <th>% Sconto</th>
                            <th>Totale</th>
                        </tr>
                        </thead>
                        <tbody>
                        @foreach($order->orderRow as $orderRow)
                            <tr>
                                <td>#{{ $orderRow->id_cfn_rig }}</td>
                                <td>{{ $orderRow->getProdDsc() }}</td>
                                <td>{{ $orderRow->qta }}</td>
                                <td>{{ $orderRow->prz_uni() }}</td>
                                <td>{{ $orderRow->sco() }}</td>
                                <td>{{ $orderRow->tot_imp() }}</td>
                            </tr>
                        @endforeach
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
@stop
```

4.4.4 Lista Anagrafiche

Codice 4.10: Lista Anagrafiche

```
Route::get('/customers', 'CustomerController@index')->name('customer.index');
```

```
/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */

public function index()
{
    $customers = new Customer;
    $customers = $customers->where('fld_custom_1','!=','')->orderBy('fld_custom_1')->get();

    return view('customer.index',compact('customers'));
}

@extends('layout.default')
@section('content')
    <div id = "container" class="container">
        <div class="row">
            <div class="col s12 l12">
                <div class="card-panel row">
                    <table class="responsive-table">
                        <thead>
                        <tr>
                            <th>ID</th>
                            <th>Ragione Sociale</th>
                        </tr>
                        </thead>
                        <tbody>
                        @foreach($customers as $customer)
                            <tr>
                                <td>
                                    <a href="{{
                                     route('customer.show',
                                     [$customer->id_ana]) }}">
                                        #{{ $customer->id_ana }}
                                    </a>
                                </td>
                                <td>
                                    <a href="{{
                                    route('customer.show',
                                     [$customer->id_ana]) }}">
                                        <span class="flag-icon flag-icon-{{
                                        strtolower($customer->getAlphaCodeNation()) }}
                                        flag-icon-squared"></span>
                                        {{ $customer->fld_custom_1 }}
                                    </a>
                                </td>
                            </tr>
                        @endforeach
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
@stop
```

4.4.5 Dettaglio Anagrafiche

Codice 4.11: Dettaglio Anagrafiche

```
Route::get('/customers/{id}', 'CustomerController@show')->name('customer.show');

/**
 * Display the specified resource.
```

```
 *
 * @param  int  $id
 * @return \Illuminate\Http\Response
 */
public function show($id)
{
   $customer = new Customer;
   $customer = $customer->find($id);

   return view('customer.show',compact('customer'));
}

@extends('layout.default')
@section('content')
    <div class="container">
        <div class="row">
            <div class="col s12 l12">
                <div class="card-panel row">
                    <h3>  {{ $customer->fld_custom_1 }} </h3>
                    <span># {{ $customer->id_ana }}</span>

                    <table class="responsive-table">
                        <thead>
                        <tr>
                            <th>Numero</th>
                            <th>Data</th>
                            <th>Totale</th>
                        </tr>
                        </thead>
                        <tbody>
                        @foreach($customer->orders as $order)
                            <tr>
                                <td>
                                <a href="{{
                                route('order.show', [$order->id_cfn]) }}">
                                #{{ $order->id_cfn }}
                                </a>
                                </td>
                                <td>
                                {{ format_it_date($order->dat_cfn) }}
                                </td>
                                <td>
                                {{ $order->totalOrder() }}
                                </td>
                            </tr>
                        @endforeach
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
@stop
```

4.4.6 Lista Nazioni

Codice 4.12: Lista Nazioni

```
Route::get('/nations', 'NationController@index')->name('nation.index');

/**
 * Display a listing of the resource.
 *
 * @return \Illuminate\Http\Response
 */
public function index()
```

```
{
    $nations = new Nation;
    $nations = $nations
    ->where('cit_nom_prv','Stato Estero')
    ->orWhere('cit_idd','999907')
    ->orderBy('cit_nom')->get();

    return view('nation.index', compact('nations'));
}

@extends('layout.default')
@section('content')
    <div id = "container" class="container">
        <div class="row">
            <div class="col s12 l12">
                <div class="card-panel row">
                    <table class="responsive-table">
                        <thead>
                        <tr>
                            <th>ID</th>
                            <th>Nome</th>
                            <th>Name (En)</th>
                            <th>Alpha Code</th>
                        </tr>
                        </thead>
                        <tbody>
                        @foreach($nations as $nation)
                            <tr>
                                <td>
                                    #{{ $nation->cit_idd }}
                                </td>
                                <td>
                                    <span class="
                                    flag-icon flag-icon-{{
                                    strtolower($nation->alpha_code_iso) }}
                                     flag-icon-squared"></span>
                                    {{ $nation->cit_nom }}
                                </td>
                                <td>
                                    {{ $nation->name }}
                                </td>
                                <td>
                                   {{ $nation->alpha_code_iso }}
                                </td>
                            </tr>
                        @endforeach
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
@stop
```

4.4.7 Layout Default

Codice 4.13: Layout Default

```
<!DOCTYPE html>
<html lang="it">
@include('include.header')
<body id="backtotop">
@yield('content')
@include('include.footer')
</body>
</html>
```

4.4.8 Header

Codice 4.14: Header

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it" lang="it">
<head>
    <!-- Primary Meta Tags -->
    <title>
    Visualizzazione di Dati di medie e piccole aziende - UniversitÃă di Padova
    </title>
    <meta name="title" content="
    Visualizzazione di Dati di medie e piccole aziende - UniversitÃă di Padova">
    <meta name="description" content="
    Visualizzazione di Dati di medie e piccole aziende. Tesi di Giacomo Fabbian.
    Laurea in Ingegneria Informatica UniversitÃă di Padova.">

    <!-- Open Graph / Facebook -->
    <meta property="og:type" content="website">
    <meta property="og:url" content="https://metatags.io/">
    <meta property="og:title"
    content="Visualizzazione di Dati di medie e piccole aziende -
    UniversitÃă di Padova">
    <meta property="og:description"
    content="Visualizzazione di Dati di medie e piccole aziende.
    Tesi di Giacomo Fabbian.
    Laurea in Ingegneria Informatica UniversitÃă di Padova.">
    <meta property="og:image" content="{{ asset('images/gf-logo.jpg') }}">

    <!-- Twitter -->
    <meta property="twitter:card" content="summary_large_image">
    <meta property="twitter:url" content="https://metatags.io/">
    <meta property="twitter:title"
    content="Visualizzazione di Dati di medie e piccole aziende
     - UniversitÃă di Padova">
    <meta property="twitter:description"
    content="Visualizzazione di Dati di medie e piccole aziende.
     Tesi di Giacomo Fabbian.
    Laurea in Ingegneria Informatica UniversitÃă di Padova.">
    <meta property="twitter:image" content="{{ asset('images/gf-logo.jpg') }}">

    <meta charset="utf-8">
    <link rel="apple-touch-icon" sizes="57x57"
    href="{{ asset('/icon/apple-icon-57x57.png') }}">
    <link rel="apple-touch-icon" sizes="60x60"
    href="{{ asset('/icon/apple-icon-60x60.png') }}">
    <link rel="apple-touch-icon" sizes="72x72"
    href="{{ asset('/icon/apple-icon-72x72.png') }}">
    <link rel="apple-touch-icon" sizes="76x76"
    href="{{ asset('/icon/apple-icon-76x76.png') }}">
    <link rel="apple-touch-icon" sizes="114x114"
    href="{{ asset('/icon/apple-icon-114x114.png') }}">
    <link rel="apple-touch-icon" sizes="120x120"
    href="{{ asset('/icon/apple-icon-120x120.png') }}">
    <link rel="apple-touch-icon" sizes="144x144"
    href="{{ asset('/icon/apple-icon-144x144.png') }}">
    <link rel="apple-touch-icon" sizes="152x152"
    href="{{ asset('/icon/apple-icon-152x152.png') }}">
    <link rel="apple-touch-icon" sizes="180x180"
    href="{{ asset('/icon/apple-icon-180x180.png') }}">
    <link rel="icon" type="image/png" sizes="192x192"
    href="{{ asset('/icon/android-icon-192x192.png') }}">
    <link rel="icon" type="image/png" sizes="32x32"
    href="{{ asset('/icon/favicon-32x32.png') }}">
    <link rel="icon" type="image/png" sizes="96x96"
    href="{{ asset('/icon/favicon-96x96.png') }}">
    <link rel="icon" type="image/png" sizes="16x16"
```

```
        href="{{ asset('/icon/favicon-16x16.png') }}">
        <link rel="manifest" href="{{ asset('/icon/manifest.json') }}">
        <meta name="msapplication-TileColor"
        content="#ffffff">
        <meta name="msapplication-TileImage"
        content="{{ asset('/icon/ms-icon-144x144.png') }}">
        <meta name="theme-color" content="#357070">
        <meta name="viewport" content="initial-scale=1.0, maximum-scale=2.0">
        <title>Tesi</title>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        {{--CSRF Token--}}
        <meta name="csrf-token" content="{{ csrf_token() }}">
        {{-- Material Icon --}}
        <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
        {{--Common App Styles--}}
        <link rel="stylesheet" href="{{ elixir("assets/css/style.min.css") }}">
        <!--Let browser know website is optimized for mobile-->
        <meta name="viewport"
        content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no"/>
        <!-- Global site tag (gtag.js) - Google Analytics -->
        <script async src="https://www.googletagmanager.com/gtag/js?id=UA-67919461-11"></script>
        <script>
            window.dataLayer = window.dataLayer || [];
            function gtag(){dataLayer.push(arguments);}
            gtag('js', new Date());

            gtag('config', 'UA-67919461-11');
        </script>

    </head>
    <body>
    <div class="navbar-fixed">

        {{--Navigation for Desktop--}}
        <nav class="" id="menu">
            <div class="nav-wrapper">
                <a href="#" data-target="slide-out" class="sidenav-trigger">
                <i class="material-icons">menu</i></a>
                <a href="{{ route('homepage') }}" id="logo_home" class="brand-logo">
                    <img src="{{ asset('images/gf-logo.jpg') }}" class=""/>
                    <span id="header-label">TESI di Giacomo Fabbian</span>
                </a>
                <ul id="nav-mobile" class="right hide-on-med-and-down">
                    <li>
                        <a href="{{ route('homepage') }}">
                            Home
                        </a>
                    </li>
                    <li>
                        <a href="{{ route('report.index') }}">
                            Report
                        </a>
                    </li>
                    <!-- Dropdown Trigger -->
                    <li>
                        <a href="{{ route('order.index') }}">
                            Ordini
                        </a>
                    </li>
                    <li>
                        <a href="{{ route('customer.index') }}">
                            Anagrafica
                        </a>
                    </li>
                    <li>
                        <a href="{{ route('nation.index') }}">
```

```
                        Nazioni
                    </a>
                </li>
            </ul>
        </div>
    </nav>
</div>
<div class="container">
    {{--Navigation for Mobile--}}
    <ul id="slide-out" class="sidenav">
        <li>
            <a href="{{ route('homepage') }}">
                Home
            </a>
        </li>
        <li>
            <a href="{{ route('report.index') }}">
                Report
            </a>
        </li>
        <!-- Dropdown Trigger -->
        <li>
            <a href="{{ route('order.index') }}">
                Ordini
            </a>
        </li>
        <li>
            <a href="{{ route('customer.index') }}">
                Anagrafica
            </a>
        </li>
        <li>
            <a href="{{ route('nation.index') }}">
                Nazioni
            </a>
        </li>
    </ul>
</div>
```

4.4.9 Footer

Codice 4.15: Footer

```
<footer class="page-footer" id="footer">
    <div class="container">
        <div class="row">
            <div class="col l2 s12">
                <img src="{{ asset('images/logo_unipd.png') }}"
                class="circle" id="logo_unipd">
            </div>
            <div class="col l4 s12">
                <h5 class="white-text">Ingegneria Informatica</h5>
                <p class="grey-text text-lighten-4">
                Universita degli studi di Padova
                </p>
            </div>
            <div class="claim-logo col l2 s12">
                <img src="{{ asset('images/logo-claim.png') }}">

            </div>
            <div class="col l4 s12">
                <h5 class="white-text">Claim B&amp;Communication</h5>
                <p class="grey-text text-lighten-4">Azienda Ospitante</p>
                <a class="claim-link" href="https://bcommunication.it/">
                    www.bcommunication.it
                </a>
```

```
                </div>
            </div>
        </div>
        </div>
        <div class="footer-copyright">
            <div class="container" style="display: inline;">
            &copy; {{ date('Y') }} Giacomo Fabbian
                <a class="grey-text text-lighten-4 right" href="#">
                </a>
            </div>
        </div>
    </footer>

    {{--Common Scripts--}}
    <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
        // Load Charts and the corechart package.
        google.charts.load('current', {'packages':['corechart']});
    </script>
    <script type="text/javascript" src="{{ elixir('assets/js/main.min.js') }}"></script>

    </body>

    </html>
```

4.4.10 Comando per generare Aziende Fittizie

Codice 4.16: Comando per generare Aziende Fittizie

```
<?php

namespace App\Console\Commands;

use App\Model\Customer;
use Faker\Factory;
use Faker\Generator;
use Illuminate\Console\Command;

class GenerateCompanyName extends Command
{
    /**
     * The name and signature of the console command.
     *
     * @var string
     */
    protected $signature = 'generate:company-name';

    /**
     * The console command description.
     *
     * @var string
     */
    protected $description = 'Genera Company Name';

    /**
     * Create a new command instance.
     *
     * @return void
     */
    public function __construct()
    {
        parent::__construct();
    }

    /**
     * Execute the console command.
```

```
     *
     * @return mixed
     */
    public function handle()
    {

        $faker_it = Factory::create('it_IT');
        $faker = Factory::create('en_US');

        $customers = Customer::all();

        foreach ($customers as $customer){
            if(!is_null($customer->id_naz) &&
                            !is_null($customer->nation)) {
                $this->line("Processing order #" .
                                $customer->id_ana);

                $companyName = getUniqueCompanyName(
                    $customer->nation->cit_nom,
                    $faker_it,
                    $faker
                );

                $customer->fld_custom_1 = $companyName;
                $customer->save();

            }else{
                $this->error('Errore, Nessuna nazione associata
                all\'anagrafica #' . $customer->id_ana);
            }
        }
    }
}
```

4.4.11 Comando per generare Lista Attributi

Codice 4.17: Comando per generare Lista Attributi

```
<?php

namespace App\Console\Commands;

use App\Model\Customer;
use App\Model\Nation;
use App\Model\Order;
use App\Model\OrderRow;
use App\Model\Product;
use Illuminate\Console\Command;

class ListAttributes extends Command
{
    /**
     * The name and signature of the console command.
     *
     * @var string
     */
    protected $signature = 'list:attributes';

    /**
     * The console command description.
     *
     * @var string
     */
    protected $description = 'Command description';

    /**
```

```
     * Create a new command instance.
     *
     * @return void
     */
    public function __construct()
    {
        parent::__construct();
    }

    /**
     * Execute the console command.
     *
     * @return mixed
     */
    public function handle()
    {
        //Customer
        $model =  Customer::all()->first();
        $attributes = $model->getAttributes();
        $out = "Customer \n\n";

        foreach ($attributes as $key => $attribute){
            $out .= $key . ", ";
        }
        $out .= "\n\n\n\n";
        //Nation
        $out .= "Nation \n\n";
        $model =  Nation::all()->first();
        $attributes = $model->getAttributes();
        foreach ($attributes as $key => $attribute){
            $out .= $key . ", ";
        }
        $out .= "\n\n\n\n";
        $out .= "Order \n\n";
        //Order
        $model =  Order::all()->first();
        $attributes = $model->getAttributes();
        foreach ($attributes as $key => $attribute){
            $out .= $key . ", ";
        }
        $out .= "\n\n\n\n";
        $out .= "OrderRaw \n\n";
        //OrderRaw
        $model =  OrderRow::all()->first();
        $attributes = $model->getAttributes();
        foreach ($attributes as $key => $attribute){
            $out .= $key . ", ";
        }
        $out .= "\n\n\n\n";
        $out .= "Product \n\n";
        //Product
        $model =  Product::all()->first();
        $attributes = $model->getAttributes();
        foreach ($attributes as $key => $attribute){
            $out .= $key . ", ";
        }

        $this->line($out);
    }
}
```

4.5 Report SQL

4.5.1 Query SQL

Codice 4.18: Query SQL

```
SELECT paese, account, SUM(round((sum_imp),2) ) as tot, n_ordini
FROM (SELECT half.cit_nom AS paese, sum_imp, n_ordini, cit_idd
FROM (SELECT cit_idd, cit_nom, SUM(imp)as sum_imp
FROM (SELECT cit_nom, id_cfn ,cit_idd
FROM (SELECT id_ana, cit_nom,cit_idd
FROM (SELECT id_ana, id_naz
FROM ana WHERE " . $cond_paese . ")as an
INNER JOIN (SELECT cit_idd, cit_nom
FROM dat_cit) as nome_cit ON id_naz = cit_idd) as ana_ok
INNER JOIN (SELECT id_cfn,id_ana
FROM cfn WHERE " . $cond_anno . ")as cfn_o
ON ana_ok.id_ana = cfn_o.id_ana) as cfn_ok
INNER JOIN cfn_rig ON cfn_ok.id_cfn = cfn_rig.id_cfn
GROUP BY cit_nom,imp)as half
INNER JOIN (SELECT cit_nom, COUNT(*) as n_ordini
FROM (SELECT id_ana, cit_nom
FROM (SELECT id_ana, id_naz
FROM ana WHERE " . $cond_paese . ")as an
INNER JOIN (SELECT cit_idd, cit_nom
FROM dat_cit) as nome_cit ON id_naz = cit_idd) as ana_ok
INNER JOIN (SELECT id_cfn,id_ana
FROM cfn WHERE " . $cond_anno . ")as cfn_oi
ON ana_ok.id_ana = cfn_oi.id_ana
GROUP BY cit_nom)as other_half
ON half.cit_nom = other_half.cit_nom) as one
INNER JOIN
(SELECT cit_nom, id_naz, COUNT(*) as account
FROM (SELECT cit_nom, id_naz, d.id_ana
FROM (SELECT cit_nom, id_naz, cfn_ot.id_ana
FROM (SELECT id_naz, cit_nom, id_ana
FROM (SELECT id_ana, id_naz
FROM ana WHERE " . $cond_paese . ")as an
INNER JOIN dat_cit ON id_naz = cit_idd)as f
INNER JOIN (SELECT id_cfn,id_ana
FROM cfn WHERE " . $cond_anno . ") as cfn_ot
ON f.id_ana = cfn_ot.id_ana)as d
GROUP BY id_ana) as r GROUP BY id_naz) as other_one
ON cit_idd = id_naz GROUP BY paese ORDER BY cit_nom;
```

Capitolo 5

Conclusioni

In questa tesi ho descritto il progetto sviluppato per Claim; per permettere di visualizzare i dati estratti dal database al cliente. Grazie a questa esperienza ho potuto rafforzare le mie conoscenze sui linguaggi di programmazione per il web e, cosa più importante, ho imparato ad usare Jquery, una libreria di JavaScript attraverso la quale si possono avviare delle chiamate Ajax. Attraverso questo tecnica di scambio dati, lo stile e l'usabilità dei siti web sono aumentati in maniera notevole. Lo sviluppo di questa applicazione web mi ha permesso di essere riconfermato all'interno dell'azienda, nell'ottica di integrare quanto fatto in molteplici sezioni del gestionale aziendale. Posso affermare che questo tirocinio è stato un buon punto di inizio della mia carriera lavorativa che mi ha permesso di iniziare a lavorare come Consulente SAP e infine per iniziare a lavorare come lavoratore autonomo presso Offline Agency.

Bibliografia

[1] PHP Storm https://www.jetbrains.com/phpstorm/features/php_code_editor.html

[2] Sistemi di Basi di Dati Fondamenti RamezElmasri,ShamkantB.Navathe: SistemidiBasidiDatiFondamenti,USA,(2011),(Pearson,Italia,2011)

[3] Git https://it.wikipedia.org/wiki/Git_(software)

[4] Git https://git-scm.com/book/it/v1/Per-Iniziare-Basi-di-Git#Quasi-Tutte-le-Operazioni-Sono-Locali

[5] Google API Chart https://en.wikipedia.org/wiki/Google_Chart_API

[6] Google API Chart https://developers.google.com/chart/

[7] Laravel https://laravel.com/

[8] Laracast https://laracasts.com

[9] Model, View, Controller https://it.wikipedia.org/wiki/Model-view-controller

[10] Erp https://www.oracle.com/it/applications/erp/what-is-erp.html

[11] Sistema Client Server https://it.wikipedia.org/wiki/Sistema_client/server

[12] Materialize CSS https://materializecss.com/

[13] Immagine ERP https://www.albaconsulting.it/erp-brescia

[14] SEM https://it.wikipedia.org/wiki/Search_engine_marketing

[15] SEO https://it.wikipedia.org/wiki/Ottimizzazione_(motori_di_ricerca)