Istituto Tecnico Industriale "G. Bearzi"
Udine
ESAME di STATO

Calendary

un progetto creato per impressionare il tempo

Alessandro Battaglia
Classe 5 INFO A - A.S. 2017/2018

Sommario

Calendary è il risultato di cinque anni di studio, iniziati all'istituto Arturo Malignani di Udine e continuati all'istituto Giacomino Bearzi di Udine.

Indice

1 Premessa

Calendary nasce come ricerca di un progetto innovativo; si è cercato di seguire le passioni e gli interessi e allo stesso tempo lasciare da parte progetti semplici e non particolarmente interessanti. Per realizzare questo è stato necessario mettersi in gioco approfondendo argomenti mai trattati prima.

2 Introduzione

Il progetto è stato realizzato da un ragazzo di 19 anni, Alessandro Battaglia, con una passione per l'informatica. La prima volta che si è cimentato in un calcolatore era in 3a media durante un corso riguardante la programmazione di una scheda elettronica dotata di microcontrollore, chiamata Arduino. Lo studio dell'informatica legato al mondo web inizia, a tutti gli effetti, durante i primi anni delle superiori con i linguaggi front-end e back-end (HTML, JavaScript, PHP, ...).

2.1 Da dove nasce l'idea?

L'idea è nata da un problema quotidiano, ma allo stesso tempo non banale. Si è, quindi, trovato un problema basilare su cui poi sviluppare un'idea più completa.

Problema Calendary nasce come risoluzione ad un problema: riuscire ad organizzare gli impegni e allo stesso tempo mantenere un bilancio personale basato sugli eventi.

Domande Per comprendere meglio di cosa si tratta basta trovare la risposta alle seguenti domande:

- È mai capitato di dover concordare gli impegni da svolgere a casa?
- È mai stato necessario organizzare i propri impegni o le proprie attività giornaliere?
- È mai capitato di dover calcolare ad esempio il costo delle lezioni di pianoforte dell'ultimo periodo?

Calendary risponde a queste e altre domande.

2.2 Obiettivo

L'obiettivo del progetto è migliorare l'organizzazione personale, della famiglia o degli utenti. Il progetto, una volta ultimato, verrà pubblicato per incrementare il numero di utenti e la richiesta di funzionalità.

3 Progetto

3.1 Descrizione del progetto

Calendary è un calendario intelligente. Sono presenti tutte le funzionalità di un classico calendario digitale, con l'aggiunta di molte altre funzionalità che semplificano la gestione degli impegni personali e di un gruppo.

3.2 Contesto

Calendary è stato realizzato in un ambiente Windows, ma attuato in ambiente Linux. Questo ha permesso l'approfondimento di entrambi i sistemi operativi sotto molti punti di vista paralleli, come ad esempio il versionamento con Git [Vedi 6.1] o la gestione dei pacchetti di Composer [Vedi 6.2].

Engine Il Server HTTP utilizzato è Apache [Vedi 6.3], il DBMS è MySQL e lo script engine è PHP.

3.3 Caratteristiche del progetto

Calendary è un normale calendario digitale al quale sono state aggiunte alcune funzionalità per renderlo innovativo. La principale feature caratterizzante di Calendary è il bilancio. Vengono calcolate le spese sommando i costi degli eventi, che possono essere filtrati specificando un periodo di riferimento oppure il nome degli eventi.

3.3.1 Evento

Caratteristiche di evento Come si può notare dalla struttura del database al punto 4.2.2, un evento associato ad un calendario ha una descrizione, può essere una nota (e quindi essere visualizzato come un evento che dura tutto il giorno), può essere personalizzato (ad esempio con il colore di sfondo o del testo) e può essere pubblico.

3.3.2 Gruppo di eventi

Il gruppo di eventi raggruppa eventi simili, come per esempio eventi che si ripetono nel tempo. Ogni volta che un evento viene modificato tutte le modifiche sono ripercosse sugli altri eventi della serie. Un gruppo di eventi può essere anche identificato come un *tag*.

La particolarità di un gruppo di eventi è che gli eventi al suo interno sono astratti, questo significa che non esistono nel database. Quando viene richiesta la presenza di eventi in un determinato periodo, se il gruppo di eventi contiene eventi che interessano il periodo selezionato, allora quegli eventi si concretizzano temporaneamente. Quando un evento della serie viene modificato, viene anche

concretizzato e associato al gruppo di eventi. Questo per evitare la duplicazione dell'evento durante la visualizzazione.

3.3.3 La newsletter

La newsletter è stata implementata fin dal principio, ma utilizzata solo verso la fine del progetto. In realtà non è altro che una lista di email opportunamente organizzata per inviare gli aggiornamenti del sistema a chi è interessato.

3.3.4 La tabella *time*

La gestione del tempo dell'evento è stata organizzata in un modo particolare. La rappresentazione grafica della tabella [Vedi 4.2.2] non è esaustiva in quanto sono stati omessi elementi non fondamentali. Per ogni istanza di *time* sono presenti anche gli attributi che specificano singolarmente anno, mese, giorno, ora e minuto. Questo per riuscire a gestire meglio la ricerca degli eventi.

4 Progettazione

4.1 Requisiti

I requisiti software si suddividono in **funzionali**, **non funzionali** e **di sistema**.

4.1.1 Requisiti funzionali

Requisiti funzionali descrivono le funzionalità che devono essere fornite dal software, di come il software reagisce a specifici tipi di input e di come si comporta in situazioni particolari. Rispondono alla domanda **"Cosa deve fare il sistema?"** e vengono descritti i **vincoli operativi**.

Utente visitatore L'*utente visitatore* dovrà poter:

- eseguire l'accesso;
- registrarsi;
- visualizzare gli eventi pubblici;
- chiedere accesso ad uno specifico calendario.

Utente registrato L'*utente registrato* dovrà poter:

- utilizzare il sistema come un *utente visitatore*
- visualizzare i calendari nella sua lista;
- aggiungere un calendario alla sua lista;
- modificare un calendario dalla sua lista;
- eliminare un calendario dalla sua lista;
- visualizzare gli eventi pubblici nelle vicinanze;
- visualizzare l'effettiva partecipazione ad un evento pubblico nelle vicinanze;
- confermare la partecipazione ad un evento pubblico nelle vicinanze;
- rifiutare la partecipazione ad un evento pubblico nelle vicinanze;
- annullare la partecipazione ad un evento pubblico nelle vicinanze;
- personalizzare un calendario di cui ha il permesso di *modifica*;
- visualizzare gli eventi di un calendario di cui ha il permesso di *visualizzazione*;
- aggiungere un evento ad un calendario di cui ha il permesso di *creazione*;

- modificare un evento di un calendario di cui ha il permesso di *modifica*;
- eliminare un evento di un calendario di cui ha il permesso di *eliminazione*;
- acquisire il permesso di *modifica* di un calendario condiviso;
- acquisire il permesso di *eliminazione* di un calendario condiviso;
- calcolare quanto manca per una scadenza;
- visualizzare degli avvisi programmati per un evento;
- aggiungere degli avvisi per un evento;
- modificare i propri avvisi ad un evento;
- eliminare i propri avvisi ad un evento;
- partecipare a uno o più gruppi di utenti (massimo 4);
- segnalare un bug;
- segnalare un evento;
- visualizzare le proprie segnalazioni.

Un evento di un *utente registrato* può contenere:

- titolo;
- luogo;
- giorno e ora;
- partecipanti;
- utenti a cui è condiviso il calendario;
- promemoria;
- note.

Utente premium L'*utente premium* dovrà poter:

- utilizzare il sistema come un *utente registrato*;
- creare infiniti gruppi di utenti;
- partecipare a uno o più gruppi di utenti;
- modificare un gruppo di utenti se ha il permesso *modifica*;
- eliminare un gruppo di utenti se ha il permesso *eliminazione*;
- acquisire il permesso di *modifica* di un calendario di un gruppo di utenti;

- acquisire il permesso di *eliminazione* di un calendario di un gruppo di utenti.

Un evento di un *utente premium* può contenere (oltre a ciò che contiene un evento di un utente registrato):

- prezzo dell'evento;
- gruppo di appartenenza dell'evento.

Utente amministratore L'*utente amministratore* dovrà poter:

- utilizzare il sistema come un *utente premium*;
- visualizzare gli errori di sistema;
- visualizzare le richieste di supporto degli utenti;
- rispondere le richieste di supporto degli utenti;
- modificare gli eventi;
- eliminare gli eventi;
- inviare una segnalazione ad un utente per violazione delle regole del sistema;
- visualizzare la lista degli utenti omettendo i dati sensibili;
- visualizzare la lista degli utenti completa inserendo due password;
- sospendere un utente con un motivo valido (tre segnalazioni ad un utente oppure almeno una violazione grave delle regole del sistema).

Sistema Il *sistema* dovrà poter:

- permettere l'utilizzo del sistema via web;
- segnalare l'errato inserimento dei dati per ogni form[1];
- segnalare la sovrapposizione di più eventi;
- segnalare all'utente un accesso sospetto, quindi da una nuova posizione (stato oppure client[2])
- inviare una notifica quando viene richiesto un avviso.

[1] **Form**: (EN) /fôrm/ – An arrangement of the elements in a composition or discourse

[2] **Client**: (EN) /'klīent/ – A person or group that uses the professional advice or services of a lawyer, accountant, advertising agency, architect, etc.

4.1.2 Requisiti non funzionali

Requisiti non funzionali descrivono le proprietà del sistema in relazione a determinati servizi o funzioni. I requisiti non funzionali vengono categorizzati in:

- usabilità (*usability*):
  - Il sistema è facile da usare?
  - Il sistema ha un'interfaccia intuitiva?
- affidabilità (*reliability*):
  - Il sistema è pronto a rispondere ad un errore?
  - Il sistema rispetta un processo esecutivo sicuro? Rispetta un ISO di un organo specifico?
- rendimento (*performance*):
  - Il sistema supporta elevate interazioni?
  - Il sistema riesce a rispondere entro un certo periodo di tempo limitato?
- manutenibilità (*supportability*):
  - Il sistema è interrogabile dalle API? È in grado di essere interrogato da altri sistemi?
  - Il sistema è suddiviso in macro sezioni? È facile aggiungere una sezione? È possibile aggiungerne una durante o a seguito della realizzazione?

Di seguito sono riportati i **requisiti non funzionali** del sistema.

- Il sistema deve preservare la segretezza dei dati personali e deve garantire che questi non siano visibili da utenti non autorizzati;
- in caso di manutenzione il sistema deve assicurarsi che sia stata esposta una richiesta ufficiale di manutenzione;
- il sistema deve garantire il completo utilizzo sia su dispositivi desktop che su dispositivi portatili con tecnologia touchscreen[3] e deve adattarsi allo schermo di qualunque dispositivo[4];
- il sistema deve garantire la risposta in un tempo minore di 4 secondi dall'inoltro della richiesta da parte di un utente;

[3] **Touchscreen**: (EN) /taCHskrēn/ – A display device that allows a user to interact with a computer by touching areas on the screen.

[4] Il sistema deve essere responsive e garantire quindi la compatibilità con la maggior parte dei dispositivi

- il sistema deve rispettare la regola dei tre tocchi, deve essere quindi possibile raggiungere ogni schermata principale del sistema in massimo 3 azioni;
- il sistema deve prevedere l'ampliamento e la modifica successiva alla consegna. Non sono permessi cambiamenti in corso d'opera;
- il sistema deve poter essere installato anche da remoto;
- il codice sorgente del sistema appartiene alla software house[5] che lo ha scritto, non è quindi di proprietà del committente;
- il sistema deve avvisare l'utente circa l'utilizzo di tutti i dati inseriti, direttamente ed indirettamente, nel sistema mediante un avviso.

Requisiti di prodotto

- Affidabilità
  - I fenomeni di riavvio del sistema dovrebbero essere meno di 1 al mese;
  - il sistema deve funzionare 24/7 fornendo, tuttavia, all'utente, supporto quotidiano, ma non continuativo[6].
- Portabilità
  - Il sistema deve essere disponibile sia per le piattaforme Windows gestite da Microsoft[7], MacOS[8], Linux[9], Android[10] e iOS[11] mediante accesso web;
  - il sistema deve funzionare su qualsiasi dispositivo e deve offrire ottime responsive skills[12].

[5] **Software house**: (EN) /'sôftwe(a)r haus/ – A company whose primary products are various forms of software, software technology, distribution, and software product development.

[6] Il supporto deve essere garantito, ma non immediato, quindi il tempo massimo di risposta deve essere di 24 ore.

[7] **Microsoft Corporation**: it is an American multinational technology company with headquarters in Redmond, Washington. It develops, manufactures, licenses, supports and sells computer software, consumer electronics, personal computers, and services.

[8] **MacOS**: is a series of graphical operating systems developed and marketed by Apple Inc. since 2001. It is the primary operating system for Apple's Mac family of computers.

[9] **Linux**: is a family of free and open-source software operating systems built around the Linux kernel. Typically, Linux is packaged in a form known as a Linux distribution for both desktop and server use

[10] **Android**: is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touchscreen mobile devices such as smartphones and tablets.

[11] **iOS**: is a mobile operating system created and developed by Apple Inc. exclusively for its hardware. It is the operating system that presently powers many of the company's mobile devices, including the iPhone, iPad, and iPod Touch.

[12] **Responsive skills**: /ri'spänsiv skils/ letteralmente "capacità di risposta"; il sistema deve essere in grado di rispondere ad una richiesta che provenga anche da altri sistemi mediante le API (Application Programming Interface).

- Efficienza
  - L'utilizzo del sistema deve comportare all'utente un consumo dati limitato;
  - il tempo di risposta non deve essere superiore al secondo.
- Performance
  - Il processo di registrazione non dovrebbe durare più di 10 minuti;
  - l'intero processo di aggiunta di informazioni nel sistema non dovrebbe durare più di 5 minuti nel 60% dei casi
- Usabilità
  - Il sistema deve essere sviluppato tramite l'approccio UCD[13], così da fornire un servizio intuitivo e di semplice utilizzo all'utente finale.
- Robustezza
  - Il sistema dovrebbe riavviarsi in meno di 10 minuti nell'80% dei casi.
- Supportabilità
  - Il sistema è supportato dal team di sviluppo fino a 1 anno alla messa in opera.

Requisiti organizzativi

- Consegna
  - Il sistema deve essere consegnato entro e non oltre la data di esposizione della tesi.
- Implementazione
  - Il fornitore deve sviluppare il sistema in modo che i requisiti restino validi anche nel futuro, fornendo la possibilità di implementarne di nuovi.

Requisiti esterni

- Interoperabilità
  - Il sistema deve essere in grado di interagire con altri sistemi già esistenti o ancora in divenire, senza restrizioni.
- Etici

[13] **UCD**: la *progettazione centrata sull'utente* (o in inglese: User-centered design (UCD)) è una filosofia di progettazione e un processo nel quale ai bisogni, ai desideri e ai limiti dell'utente sul prodotto finale è data grande attenzione in ogni passo del processo di progettazione per massimizzare l'usabilità del prodotto stesso. (Cit. `https://wikipedia.org`)

  - Il sistema deve garantire la sicurezza e la salute degli utenti.

- Legislativi
  - Il sistema deve funzionare rispettando le norme e la legislatura dello stato in cui viene utilizzato.

4.1.3 Requisiti di dominio

Di seguito sono riportati i **requisiti di dominio** del sistema.

- Il sistema deve supportare tutte le sue funzionalità su tutti i dispositivi sui quali è installato;
- il sistema deve rispondere ad una richiesta in un tempo inferiore ai 2 secondi nel 98% dei casi;
- il sistema deve utilizzare al massimo 5Mb di Ram per richiesta
- il sistema deve garantire il corretto inserimento di una riga nel database; nel caso in cui la riga non sia completa deve segnalare l'errore ed eventualmente modificarla o eliminarla;
- il sistema deve rispettare la regola dei tre tocchi, deve essere quindi possibile raggiungere ogni schermata principale del sistema in massimo 3 azioni;
- il sistema deve essere protetto da una password alfanumerica che include almeno un numero, una lettera maiuscola e una minuscola;
- il sistema deve poter essere modificato anche da remoto.

4.2 Database

"A database is an organized collection of data."

— **New I-Tech** - Pag 116
English for Information and Communications Technology

4.2.1 Cenni teorici

Tipi di database Sono presenti vari tipi di database.

Database relazionale Il *database relazionale* è caratterizzato da più tabelle collegate tra loro da una chiave.

Class
id (int)
name (varchar)
location (varchar)
school (varchar)

Student
id (int)
class (int)
name (varchar)
surname (varchar)

L'attributo *class* in **Student** è chiave esterna e fa riferimento alla chiave primaria di **Class**. Questo significa che le due tabelle sono collegate tra loro.

Database NoSql Il *database NoSql* (Not Only Sql) è caratterizzato dalla possibilità di avere righe con numero di colonne differenti.

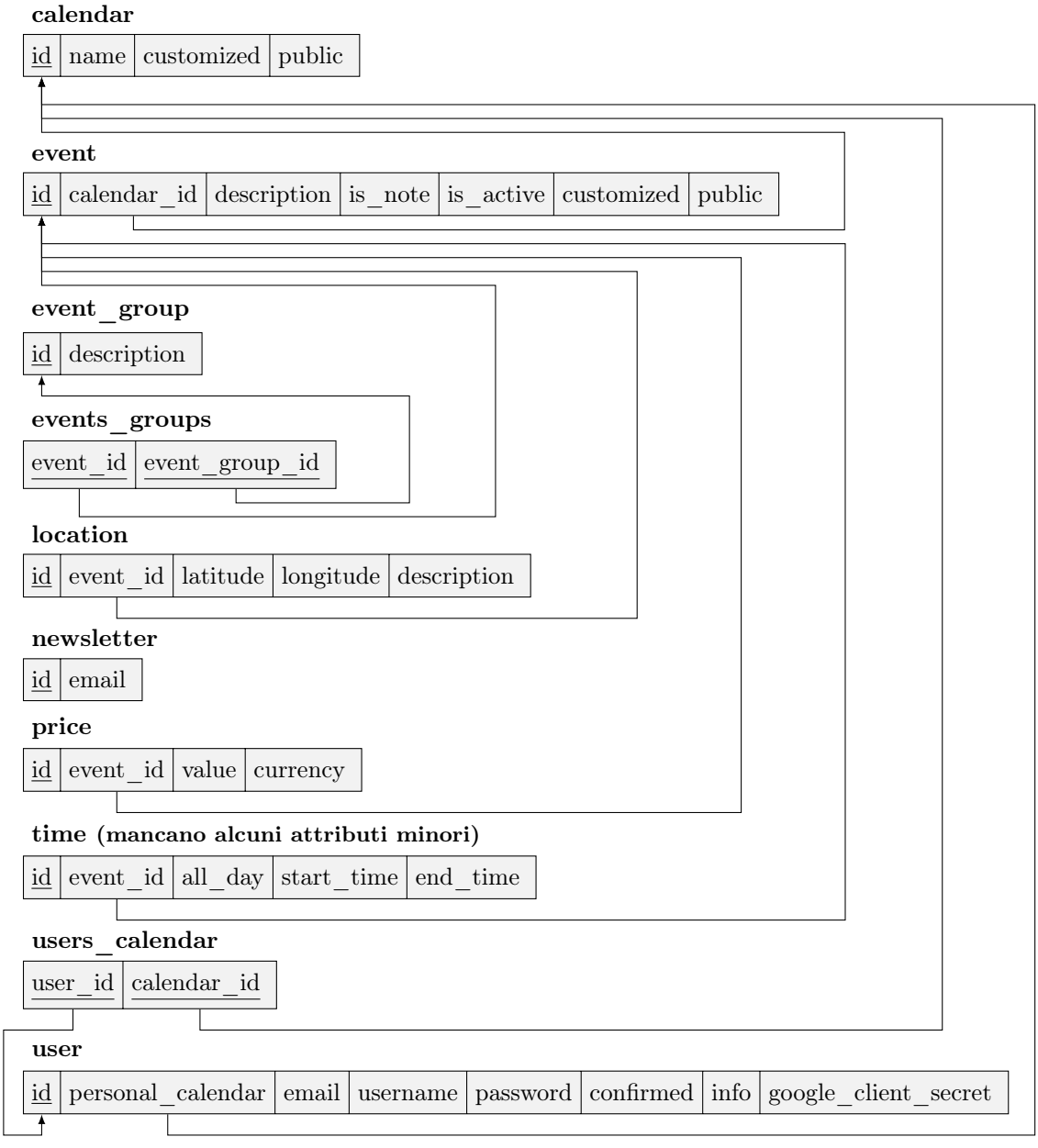
Student			
id	**name**	**surname**	**age**
1	John	Smith	
2	Mark	Addson	8 years

Le due istanze del database hanno completi diversi attributi. Dello studente con id 1 (John Smith) si conoscono solo *nome* e *cognome*; lo studente con id 2 (Mark Addson, 8 years) ha noto anche l'*età*.

Ogni istanza di tabella sfrutta la rindondanza per salvare le informazioni. La struttura tipica di un database NoSql è simile ad un file JSON, in cui il numero degli elementi di un array non è predefinito a priori.

4.2.2 Il database

Di seguito la struttura del database del progetto:



Ultimo aggiornamento 13/06/2018

5 La realizzazione

5.1 Software e tools utilizzati

JetBrains programs L'ambiente di programmazione utilizzato è PhpStorm, la visualizzazione del database è stata effettuata utilizzando DataStorm.

Git Il versionamento è stato gestito con Git [Vedi 6.1].

Composer I pacchetti sono stati gestiti con Composer [Vedi 6.2].

Jenkins L'aggiornamento del codice nel server di produzione è stato eseguito da Jenkins.

Bootstrap La struttura di una pagina web realizzata in HTML deve essere formattata. Viene utilizzato Bootstrap 4, un framework CSS che sfrutta le nuove funzionalità proposte da CSS3.

5.2 Inizializzazione

Per prima cosa è stato inizializzato il progetto Symfony [Vedi 6.4] con il versionamento di Git.

Il passo successivo è stato quello di creare una struttura del codice tale da rendere l'intero progetto facilmente modificabile anche a distanza di tempo.
[Edit] A posteriori questo passo ha portato ad una facile manutenzione e ad un veloce riambientamento dopo un periodo di pausa dovuto allo studio scolastico.

Il passo più importante è stato stendere la lista delle funzionalità da realizzare, dando una priorità agli elementi della lsta e creando *todo* ovunque nel codice.

Prima di iniziare è stato realizzato un questionario in cui venivano richieste alcune informazioni come:

- quanto è importante la grafica in un calendario (punteggio medio di 8,12 in una scala da 1 a 10);
- quanto spesso viene usato il calendario (in un totale di 41 risposte i risultati sono stati: 18 ogni giorno, 11 qualche volta alla settimana, 4 una volta alla settimana 6 poche volte al mese e 2 raramente in un anno);
- per cosa viene usato il calendario (le risposte più votate sono state "per lavoro" e "per impegni personali");
- cosa manca in un calendario (sono state proposte molte funzionalità, alcune sono state prese in considerazione (come il meteo e il lunario), altre sono state scartate perché non molto esplicative);

- l'indirizzo email per una sottoscrizione facoltativa ad una newsletter (alcuni hanno lasciato un indirizzo).

5.3 Stesura del codice

La stesura del codice è fondamentale, ogni riga del codice deve essere opportunamente scritta per evitare di dover decifrare il significato.

Durante la creazione del codice è stato fondamentale ridurre il più possibile i commenti creando moltissime funzioni che contenessero massimo 10 righe ciascuna. Questo per questioni di manutenibilità e di leggibilità del codice.

La lista delle funzionalità da implementare ha costretto a seguire una scaletta rigida. Se durante la realizzazione venivano pensate nuove funzionalità, queste venivano inserite nella lista in base alle priorità in modo da non perdere il ritmo di programmazione.

5.4 Problemi riscontrati

Il principale problema è stato ambientarsi in Symfony [Vedi 6.4], un framework che richiede conoscenze approfondite in moltissimi linguaggi come twig [Vedi 6.4.5 paragrafo "The language twig"] e in molti pacchetti PHP come Doctrine [Vedi 6.4.4] che permette l'iterazione con il database utilizzando una sintassi personalizzata.

Un problema comune alla realizzazione di diversi progetti è quello di non cadere nella tentazione di riscrivere completamente tutto il codice quando viene scoperta una nuova estensione (come un Bundle [Vedi 6.4.1 verso la fine]) che semplifica la realizzazione di una funzionalità. Resistere non è sempre facile perché spesso il codice risulta sporco e di difficile comprensione. La cosa migliore è, comunque, riscrivere il codice utilizzando una scrittura più pulita.

5.4.1 Cose da tenere da conto durante la realizzazione

Durante la realizzazione è fondamentale tenere in considerazione la compatibilità con l'ambiente mobile, sempre più utilizzato dall'utente medio.

5.5 To do list - cose da fare

Le funzionalità da implementare sono molte, come per esempio:

- la gestione delle notifiche, ignorata perché considerata di minore importanza;
- la modifica diretta degli eventi di Google Calendar non considerata fondamentale per il funzionamento del progetto;
- ulteriori funzionalità saranno aggiunte in base alle richieste degli utenti.

6 Approfondimenti

In questa sezione saranno esposti alcuni approfondimenti su programmi o tecnologie utilizzate.

6.1 Git

"Git è un software di controllo versione distribuito utilizzabile da interfaccia a riga di comando, creato da Linus Torvalds nel 2005."

— **Wikipedia, l'enciclopedia libera**
`https://it.wikipedia.org/wiki/Git_(software)`

6.1.1 Cos'è Git?

Git è un software che permette la gestione del versionamento di un progetto. Ogni versione viene salvata in una repository. La repository può essere ospitata in un server privato o in uno pubblico. Per il progetto è stato scelto un servizio pubblico (`https://bitbucket.org/`) mantenendo una versione sincronizzata anche in un server privato per praticità.

6.1.2 Perché usare Git?

Git è un software creato per ottimizzare la sincronizzazione dei progetti. Utilizzare Git significa organizzare la coordinazione del progetto intero. Se non lo si sceglie, si utilizza sicuramente un sistema di controllo versione differente (come CVS, Subversion, Perforce, Bazaar, ...).

6.1.3 Come funziona Git?

Git ha un funzionamento complesso che riesce a gestire il versionamento di file grazie ad alcune features[14] tra cui:

i branch che permettono la suddivisione del codice in diverse sezioni che possono essere lavorate in contemporanea.

Ulteriori informazioni sul software Git possono essere visionate al seguente link: `https://git-scm.com/book/en/v2`.

[14] **Features**: (EN) /fee-cher/ – A prominent or conspicuous part or characteristic

6.2 Composer

"Composer is a tool for dependency management in PHP. It allows you to declare the libraries your project depends on and it will manage (install/update) them for you."

— **Composer website** `https://getcomposer.org`

6.2.1 Cos'è composer?

Composer è un gestore di pacchetti per PHP costituite da librerie; queste possono essere sincronizzate utilizzando repository Git [Vedi 6.1] private o pubbliche oppure usando dei registri pubblici offerti da `https://packagist.org/`

6.2.2 Come funziona?

Il funzionamento è semplice : tutte le librerie scaricate vengono automaticamente rese disponibili al programmatore, l'unica accortezza da tenere in considerazione è quella di includere all'interno del proprio codice il file di Composer. Lo script engine si occupa del resto. PhpStorm [Vedi 5.1] supporta il funzionamento di Composer e semplifica la scrittura del codice.

6.3 Apache

"Apache HTTP Server, o più comunemente Apache, è il nome di un server web libero sviluppato dalla Apache Software Foundation. È la piattaforma server Web modulare più diffusa, in grado di operare su una grande varietà di sistemi operativi, tra cui UNIX/Linux, Microsoft Windows e OpenVMS."

— **Wikipedia, l'enciclopedia libera**
https://it.wikipedia.org/wiki/Apache_HTTP_Server

6.3.1 Cos'è Apache?

Apache è un server web che gestisce lo spazio web organizzandolo in server virtuali.

Di seguito un esempio di configurazione di un server virtuale:

```
<VirtualHost *:80>
        ServerName calendary.it

        ServerAdmin progettialessandro@gmail.com
        DocumentRoot /home/www/calendary.it/web/web

        <Directory /home/www/calendary.it/web/web>
                Options FollowSymLinks MultiViews
                AllowOverride All
                Require all granted
        </Directory>

        CustomLog /home/www/calendary.it/log/calendary.it-access.log combined
        ErrorLog /home/www/calendary.it/log/calendary.it-error.log
</VirtualHost>
```

Ulteriori informazioni possono essere trovate sul sito del software, all'indirizzo: https://httpd.apache.org/.

6.4 Symfony (EN)

"Symfony is a set of reusable PHP components and a PHP framework for web projects."

— https://symfony.com/

6.4.1 What is it?

Symfony is a set of tools which help web masters to build a website system easily. It is a back-end framework which means that it speeds up the creation and maintenance of web applications and replaces repetitive coding tasks using a bytecode cache.

Symfony manages the development phase and the final release. The programmer is able to debug a project in a simple way, and at the same time the project can be shared with users in a stable release.

The framework can be fully personalised using *Bundles*. They are additional packets which offer useful features.

6.4.2 Why you should use it

Symfony helps back-end developers to organise the structure of a project. It is suggested to use when the there are lots of pages with the same structure and lots of user interactions. The best example is a blog. There are four main pages (homepage, post page, user page, search page). Each page has a base structure in which some components are added.

6.4.3 Why you should not use it

Symfony requires PHP 5.3.3 or major, composer and the access to the server console; if one of these requirements is not satisfied, it is suggested not to use Symfony. The framework also needs a lot of RAM to run, especially in the debug mode. It means that the developer should have a performing computer.

6.4.4 Why I use it

Symfony makes everything easier. For example in order to create a new database table, it is sufficient to create a new PHP class; this class becomes an Entity and you can use it to work with the database. This functionality is offered by *Doctrine*.

6.4.5 How it works

Symfony has characteristics which make the programming easy.

The structure of a project When you initialise a new empty Symfony project (3.4.x) the structure of the page is something like the following:

```
root (the project root)
├─ app
│  ├─ config (in this directory there are config files)
│  │  └─ ...
│  └─ Resources (in this directory there are resources)
│     └─ views (in this directory there are twig files)
│        └─ ...
├─ bin (in this directory there are the developer scripts)
│  └─ ...
├─ src (in this directory there are developer PHP scripts)
│  └─ AppBundle
│     └─ Controller (in this directory there are listeners to the links)
│        └─ ...
├─ vendor (this directory is managed by Composer)
│  └─ ...
├─ web (this is the public directory -- the domain points to this directory)
│  └─ ...
├─ .gitignore
├─ composer.json
└─ README.md
```

The language twig

> "The flexible, fast, and secure template engine for PHP."
>
> — https://twig.symfony.com/

What is twig

Twig is a template engine used by Symfony to build pages. The power of twig is that its work is to fill the gaps of the source code without creating the whole structure every time.

Example Here is an example (∼\app\Resources\views\components\form.html.twig):

```
{% extends 'base.html.twig' %}

{% block body %}
    <div class="container">
        <div class="row">
            <div class="col-12 col-md-8 offset-md-2 bg-white rounded py-3 mt-2">
                <div class="px-3 bg-white rounded box-shadow">
                    {{ form_start(form) }}
                    {{ form_widget(form) }}
                    {{ form_end(form) }}
                </div>
            </div>
        </div>
    </div>
{% endblock %}
{% block stylesheets %}
    {% stylesheets
        'assets/bootstrap-datepicker/css/bootstrap-datepicker3.min.css'
    %}
        <link rel="stylesheet" type="text/css" href="{{ asset_url }}"/>
    {% endstylesheets %}
{% endblock %}
{% block javascripts %}
    {% javascripts
        'assets/bootstrap-datepicker/js/bootstrap-datepicker.min.js'
    %}
        <script type="text/javascript" src="{{ asset_url }}"></script>
    {% endjavascripts %}
{% endblock %}
```

The previous source is the basic structure of a form page made up by using the twig language.

6.4.6 Hello World with Symfony

In order to create a Symfony project the programmer has to download the Symfony installer, a file which downloads a script that is able to create a symfony project.

A project is created by typing:

symfony new <project name> <version (e.g. 3.4)>.

Reference : `https://symfony.com/download`

7 Conclusione

Il progetto ha richiesto un costante approfondimento dell'informatica, ha aumentato la soglia di attenzione e incrementato la sopportazione di un impegno costante verso la scrittura di codice.

È stato compreso come realizzare un intero progetto, senza avere conoscenze iniziali specifiche. Per un programmatore è fondamentale essere in grado di trovare una soluzione ad un problema nel minor tempo possibile per riuscire a concludere un progetto entro i tempi stabiliti.

La pressione della scadenza ha portato ad immedesimarsi ancora di più nell'ambiente, che simula (sebbene solo in parte) un lavoro nel settore informatico.

7.1 Disponibilità del progetto

Calendary è disponibile all'indirizzo `https://calendary.it/`.

7.2 Ringraziamenti

Desidero ricordare tutti coloro che mi hanno aiutato nella stesura con suggerimenti, critiche ed osservazioni: a loro va la mia gratitudine. Ricordo anche coloro che mi hanno dato idee nell'inizializzazione del progetto, rispondendo al questionario proposto.

Ringrazio anzitutto i docenti che mi hanno sostenuto e motivato durante il mio percorso di studi. È doveroso citare i prof.ri Dereani Luciano, docente di Informatica, e Tirelli Andrea, docente di tecnologie tecniche, dell'Istituto Arturo Malignani di Udine e Vassena Luca, docente di informatica, dell'Istituto Giacomino Bearzi che mi hanno supportato ed incentivato nella conoscenza dell'informatica.

Un ringraziamento particolare va a tutti coloro che mi sono stati vicini durante il mio percorso formativo, anche al di fuori dell'ambiente scolastico.

L'ultimo ringraziamento va dedicato alla mia determinazione e tenacia nel riuscire a costruire un progetto e portarlo a termine.