

**ISTITUTO TECNICO INDUSTRIALE STATALE "MAX PLANCK"**

*VIA FRANCHINI, 1 31020 - LANCENIGO DI VILLORBA (TV)*

*ORGANISMO DI FORMAZIONE ACCREDITATO PRESSO LA REGIONE VENETO: COD. N. 218*



# Syncbook



Basso Enrico  
Longhin Federico

Classe: 5BII

Anno Scolastico: 2014/2015



<b><u>1.INTRODUZIONE .....</u></b>	<b><u>1</u></b>
1.1 vCARD.....	1
1.2 CARDDAV .....	1
<b><u>2 PROBLEMA .....</u></b>	<b><u>1</u></b>
2.1 DATI DA SALVARE .....	1
2.2 DATI SALVATI SECONDO STANDARD .....	1
2.3 DATI SALVATI IN APPLE CONTACTS .....	2
2.4 DATI SALVATI IN GOOGLE CONTACTS .....	2
<b><u>3 FINALITÀ DEL PROGETTO.....</u></b>	<b><u>3</u></b>
<b><u>4 OBIETTIVI DEL PROGETTO.....</u></b>	<b><u>3</u></b>
<b><u>5 PERCHÉ UN'APPLICAZIONE WEB? .....</u></b>	<b><u>3</u></b>
<b><u>6 STRUMENTI.....</u></b>	<b><u>4</u></b>
6.1 HUGE .....	4
6.1.1 STRUTTURA E FUNZIONAMENTO .....	4
6.1.2 IL DATABASE .....	8
6.1.3 DA HUGE A SYNCBOOK .....	9
6.2 SABRE.IO .....	14
6.2.1 CREAZIONE DI UNA vCARD .....	16
6.2.2 ESEMPIO DI INTERFACCIAMENTO TRA SABREDAV E HUGE .....	19
6.3 BOOTSTRAP .....	20
6.3.1 ESEMPIO DI UTILIZZO DI BOOTSTRAP FINESTRA DI LOG-IN .....	21
6.4 JQUERY .....	23
6.4.1 ESEMPIO DI UTILIZZO DELLA FUNZIONE \$.AJAX() .....	24
6.5 REDBEAN PHP .....	25
6.5.1 ORM.....	25
6.6 GIT .....	27
6.6.1 COMMIT .....	27
6.6.2 BRANCH.....	27
6.6.3 GITHUB.....	27
6.7 SERVER .....	32
<b><u>7 CONCLUSIONI .....</u></b>	<b><u>33</u></b>
7.1 PROBLEMATICHE .....	33
7.2 IMPLEMENTAZIONI FUTURE .....	34
7.2.1 GESTIONE DI PIÙ RUBRICHE.....	34
7.2.2 IMPORTAZIONE ED ESPORTAZIONE DI vCARD .....	34
<b><u>8 BIBLIOGRAFIA .....</u></b>	<b><u>35</u></b>

<b>8.1</b>	<b>DOCUMENTAZIONE TECNICA .....</b>	<b>35</b>
8.1.1	RFC .....	35
<b>8.2</b>	<b>SITI DI RIFERIMENTO .....</b>	<b>35</b>
<b><u>9</u></b>	<b><u>SI RINGRAZIANO .....</u></b>	<b><u>35</u></b>

## 1. Introduzione

Da sempre il grande problema delle rubriche in formato digitale è la loro condivisione ed esportazione tra dispositivi diversi tra loro, questo perché le grandi aziende non si sono mai preoccupate di utilizzare uno standard comune durante il salvataggio dei contatti pensando solo ai propri interessi. In realtà degli standard per la formattazione dei dati esistono ma non sono implementati nei più comuni applicativi.

### 1.1 vCard

Il formato più utilizzato per rappresentare un contatto sotto forma digitale è la vCard, giunta alla sua quarta versione. Lo standard di rappresentazione dei dati presenti all'interno di questo file di testo è descritto nell'[RFC 6350](#). Resta, però, quasi impossibile, trovare online un servizio che rispetti le regole che vengono definite.

### 1.2 CardDAV

Il protocollo più utilizzato per la gestione di contatti all'interno del web è il CardDAV. Essendo basato sulla sintassi specifica delle vCard, CardDAV permette un semplice scambio di informazioni tra server e client durante tutte le operazioni di creazione, modifica ed eliminazione di un file vCard.

## 2 Problema

### 2.1 Dati da salvare

```
Jhon Smith  
smith.jhon@gmail.com  
+44 20 1234 5678  
1 Trafalgar Square, WC2N London, United Kingdom  
Web: http://jhonsmith.com  
Skype: jhon.smith  
Twitter: @jhonsmith
```

### 2.2 Dati salvati secondo standard

```
BEGIN:VCARD  
VERSION:4.0  
N:Smith;Jhon;;;  
FN:Jhon Smith  
EMAIL;TYPE=HOME;PREF=1:smith.jhon@gmail.com  
TEL;TYPE="CELL,HOME";PREF=1:tel:+44 20 1234 5678  
ADR;TYPE=HOME;PREF=1;;;1 Trafalgar Square;London;;WC2N;United Kingdom  
URL;TYPE=HOME;PREF=1:http://jhonsmith.com  
IMPP;TYPE=HOME;PREF=1:SKYPE:jhon.smith  
X-SOCIALPROFILE;TYPE=HOME;PREF=1:TWITTER:http://twitter.com/jhonsmith  
END:VCARD
```

Per quanto strano possa sembrare, nonostante la loro quarta versione, le vCard non supportano quelli che sono oramai diventati degli strumenti alla base delle società moderna, i social network. Questi campi vengono quindi definiti come non-standard e vanno rappresentati da una lettera “X”, seguita dal simbolo minore “-” e dal nome del campo stesso. I campi così descritti possono non essere supportati da tutte le applicazioni client attraverso le quali viene poi data la possibilità all'utente di gestire la propria rubrica.

## 2.3 Dati salvati in Apple Contacts

```
BEGIN:VCARD
VERSION:3.0
N:Smith;Jhon;;;
FN:Jhon Smith
EMAIL;type=INTERNET;type=HOME;type=pref:smith.jhon@gmail.com
TEL;type=CELL;type=VOICE;type=pref:+44 20 1234 5678
ADR;type=HOME;type=pref;;;1 Trafalgar Square;London;;WC2N;United Kingdom
item1.URL;type=pref:http://jhonsmith.com
item1.X-ABLabel:__$!<HomePage>!$__
IMPP;X-SERVICE-TYPE=Skype;type=HOME;type=pref:skype:jhon.smith
X-SOCIALPROFILE;type=twitter:http://twitter.com/jhonsmith
END:VCARD
```

Utilizzando un formato non più supportato nel mondo delle vCard, la Apple riesce a rappresentare una serie di semplici dati in un modo molto differente rispetto al precedente rendendo complessa e laboriosa l'esportazione di questa vCard in un dispositivo non appartenente alla famosa azienda statunitense. Le differenze più evidenti si possono notare nella rappresentazione del campo “URL” che viene ora raggruppato con la proprietà non-standard “X-ABLabel”. Questo metodo viene utilizzato per risolvere il problema di mostrare all'utente un sito web come proprio del contatto che egli sta visualizzando.

## 2.4 Dati salvati in Google Contacts

```
BEGIN:VCARD
VERSION:3.0
N:Smith;Jhon;;;
FN:Jhon Smith
EMAIL;TYPE=INTERNET;TYPE=HOME:smith.jhon@gmail.com
TEL;TYPE=CELL:+44 20 1234 5678
ADR;TYPE=HOME;;;1 Trafalgar Square;London;;WC2N;United Kingdom
item1.URL:http\://jhonsmith.com
item1.X-ABLabel:__$!<HomePage>!$__
X-SKYPE:jhon.smith
item2.URL:http\://twitter.com/jhonsmith
item2.X-ABLabel:Twitter
END:VCARD
```

Google Contacts non dando la possibilità all'utente di salvare all'interno delle informazioni di un contatto quelle riguardanti i social network, è necessario utilizzare la proprietà “URL” per gestirli. L'utilizzo di questa peculiare formattazione, unita a degli inutili caratteri di *escape* usati per rappresentare gli indirizzi *HTTP*, rende impossibile mantenere i medesimi dati in fase di importazione della vCard in un dispositivo che non possieda un sistema operativo Android

### 3 Finalità del progetto

In seguito a un'analisi approfondita della situazione attuale del non utilizzo degli standard vCard, appare evidente che l'importazione ed esportazione di contatti tra dispositivi diversi non può avvenire, se non in modo incompleto o corrotto. Per questo motivo si è deciso di sviluppare un progetto per dimostrare come sia possibile utilizzare gli Standard per la formattazione di una vCard traendone vantaggio durante il processo di gestione di una rubrica.

### 4 Obiettivi del progetto

Realizzare un'applicazione web che permetta, all'utente correttamente registrato, di creare delle rubriche secondo gli standard vCard. L'applicativo dovrà essere in grado di salvare i dati inseriti dall'utente e di metterli a disposizione anche ad altri dispositivi che si interfacciano tramite il protocollo WebDAV. Risulta dunque necessaria la configurazione di un server per sostenere tutte le tipologie di richieste: dal salvataggio dei dati tramite database, alla sicurezza delle comunicazioni con il client.

### 5 Perché un'applicazione web?

La scelta di sviluppare un'applicazione web è stata effettuata per diversi motivi.

Il primo è legato alla natura della libreria Sabre.io: essa infatti è sviluppata totalmente in linguaggio PHP e per questa ragione è sembrato ovvio continuare con questa tecnologia integrando nuovi servizi e funzionalità.

Il secondo motivo si basa sulla portabilità dell'applicazione: se da un lato sarebbe stato possibile progettare un'applicazione rivolta alle funzionalità lato client, come un'applet Java od un'app Android, dall'altro si è capito che un'applicazione web avrebbe avuto un maggior bacino di utenza rispetto alle altre alternative. Questo perché una piattaforma web non ha bisogno di essere installata fisicamente nel dispositivo essendo, quindi, vincolata al software del sistema operativo ospitante.

## 6 Strumenti

### 6.1 HUGE

Per realizzare le funzionalità di log-in e gestione degli utenti è stato necessario scegliere un framework PHP che implementasse già questi algoritmi in modo da concentrare lo sviluppo su quello che effettivamente veniva richiesto dagli obiettivi del progetto. Per questo motivo la scelta è ricaduta su HUGE.

A differenza di altri famosi framework, come ad esempio Symfony, HUGE è una soluzione più semplice, sia a livello di comprensione del codice, che a livello di struttura e usabilità.

HUGE è l'ultima versione di una serie di progetti PHP sviluppati da *Panique* che hanno come finalità quella di fornire un container per la fase di avvio di nuove applicazioni, cioè preparare degli strumenti funzionanti "out of the box" come la gestione degli utenti (ad esempio: registrazione e log-in) e il motore grafico.

Alcune delle funzionalità più importanti di HUGE sono:

- gli utenti possono registrarsi ed effettuare log-in e log-out;
- salvataggio password secondo gli standard ufficiali PHP (algoritmo *bcrypt*);
- password dimenticata e conseguente possibilità di reset;
- verifica dell'account via e-mail;
- supporto nativo all'invio delle e-mail (tramite PHP Mailer);
- URL rewriting.

#### 6.1.1 Struttura e funzionamento

HUGE presenta una struttura MVC, anche se i metodi utilizzati sono per lo più statici. La programmazione Model-View-Controller viene sfruttata appieno dalla configurazione di un file *.htaccess* e dal file *Application.php*. Tutte le richieste HTTP vengono reindirizzate da Apache nella cartella *public* in cui è presente il file *index.php* il cui scopo è creare una nuova istanza della classe *Application*.



#### 6.1.1.1 Il file *.htaccess* della cartella *public*

```
# Necessario per evitare problemi nel caso in cui si usi un controller
# chiamato "index" avendo un file root index.php.
# http://httpd.apache.org/docs/2.2/content-negotiation.html
Options -MultiViews

# Attivazione URL rewriting (ES.: myproject.com/controller/action/1/2/3).
RewriteEngine On

# Impedisce che si possa navigare direttamente nelle cartelle.
Options -Indexes

# Se le seguenti condizioni sono vere, allora riscrive la URL:
# Se il file richiesto non è una directory,
RewriteCond %{REQUEST_FILENAME} !-d
# e se il file richiesto non è un file esistente,
RewriteCond %{REQUEST_FILENAME} !-f
# e se il file richiesto non è un link simbolico,
RewriteCond %{REQUEST_FILENAME} !-l
# allora riscrivi la URL nel modo seguente:
# Prende l'intero nome del file richiesto e lo ritorna come valore di
# parametro "url" a index.php. Aggiunge in coda le rimanenti query string
# dell'URL originale come parametri successivi (QSA), ferma l'esecuzione di
# questo file .htaccess (L).
RewriteRule ^(.+)$ index.php?url=$1 [QSA,L]
```

Il file *.htaccess* appena descritto è presente nella cartella *public* di HUGE, allo stesso livello del file *index.php*. Questo file permette il funzionamento dello *url rewriting* e dell'esecuzione di controller/metodi tramite determinati parametri presenti nelle richieste HTTP.

### 6.1.1.2 La classe *Application.php*

```
<?php

class Application
{
    /** @var mixed Istanza del controller. */
    private $controller;

    private $parameters = array();

    /** @var string Nome del controller. */
    private $controller_name;

    /** @var string Nome del metodo del controller. */
    private $action_name;

    /**
     * Esegue lo start dell'applicazione, analizza gli elementi della URL,
     * esegue il controller e metodo corrispondenti o reindirizza a index.
     */
    public function __construct()
    {
        // Crea array con i parametri in URL in $url.
        $this->splitUrl();

        // Esegue controllo sul controller: non è stato dato un controller ?
        // allora setta ontroller = default controller (dalla configurazione).
        if (!$this->controller_name) {
            $this->controller_name = Config::get('DEFAULT_CONTROLLER');
        }

        // Controllo per action (metodo): non è stato dato una action ?
        // allora setta action = default action (dalla configurazione).
        if (!$this->action_name OR (strlen($this->action_name) == 0)) {
            $this->action_name = Config::get('DEFAULT_ACTION');
        }

        // Rinomina controller_name al reale nome controller class/file (da "index" a "IndexController").
        $this->controller_name = ucwords($this->controller_name) . 'Controller';

        // Il controller selezionato esiste?
        if (file_exists(Config::get('PATH_CONTROLLER') . $this->controller_name . '.php')) {

            // Carica questo file e crea il controller.
            // Esempio: se il controlelr fosse "car", allora si tradurrebbe in: $this->car = new car();
            require Config::get('PATH_CONTROLLER') . $this->controller_name . '.php';
            $this->controller = new $this->controller_name();

            // Controllo per il metodo: esiste all'interno del controller?
            if (method_exists($this->controller, $this->action_name)) {
                if (!empty($this->parameters)) {
                    // Chiama il metodo e gli passa i parametri.
                    call_user_func_array(array($this->controller, $this->action_name), $this->parameters)
                }
            }
        }
    }
}
```

```

        } else {
            // Se non ci sono parametri, richiama il metodo senza parametri,
            // ad esempio $this->index->index().
            $this->controller->{$this->action_name}();
        }
    } else {
        header('location: ' . Config::get('URL') . 'error');
    }
} else {
    header('location: ' . Config::get('URL') . 'error');
}
}

/**
 * Get and split the URL.
 */
private function splitUrl()
{
    if (Request::get('url')) {

        // Split URL.
        $url = trim(Request::get('url'), '/');
        $url = filter_var($url, FILTER_SANITIZE_URL);
        $url = explode('/', $url);

        // Mette i parametri URL nei rispettivi attributi di classe.
        $this->controller_name = isset($url[0]) ? $url[0] : null;
        $this->action_name = isset($url[1]) ? $url[1] : null;

        // Rimuove controller_name e action_name dall'array.
        unset($url[0], $url[1]);

        // Salva i parametri URL nell'attributo.
        $this->parameters = array_values($url);
    }
}
}

?>

```

La classe Application.php sta alla base di tutto il funzionamento dell'applicazione. Tutte le richieste al server vengono effettuate attraverso lo schema <dominio>/controller/metodo e il protocollo HTTP aggiunge gli eventuali parametri provenienti dal form. In questo modo il file .htaccess può effettuare l'operazione di *url rewriting* mostrando la stringa sopra citata e non il file che viene eseguito (in questo caso index.php).

La prima operazione effettuata nel costruttore della classe Application è l'esecuzione del metodo *splitUrl()* che setta a dei valori iniziali gli attributi della classe *controller\_name*, *action\_name* e *parameters*, che rispettivamente sono il controller e il metodo (più i parametri) che andranno poi eseguiti.

In seguito si effettuano dei controlli se esistono o meno il controller e il metodo selezionati, nel caso contrario vengono settati a valori di default (ad esempio una configurazione di default potrebbe essere *index/index*, cioè effettuare il redirect alla pagina iniziale). Una volta effettuata la convalida si procede con la creazione dell'istanza del controller selezionato e si richiama il metodo associato passando i parametri ricavati (se ce ne sono).

### 6.1.2 Il database

Per la memorizzazione dei dati degli utenti, HUGE presenta una base di dati con una tabella *users* e una tabella *notes*. La prima contiene tutte le informazioni degli utenti registrati, mentre la seconda serve solo per eseguire delle dimostrazioni di utilizzo di HUGE e non verrà utilizzata in Syncbook. La tabella *users* è stata modificata inserendo anche i campi *first\_name* e *last\_name*, necessari al completamento delle credenziali per la tabella relativa al salvataggio dell'utente in Sabre.

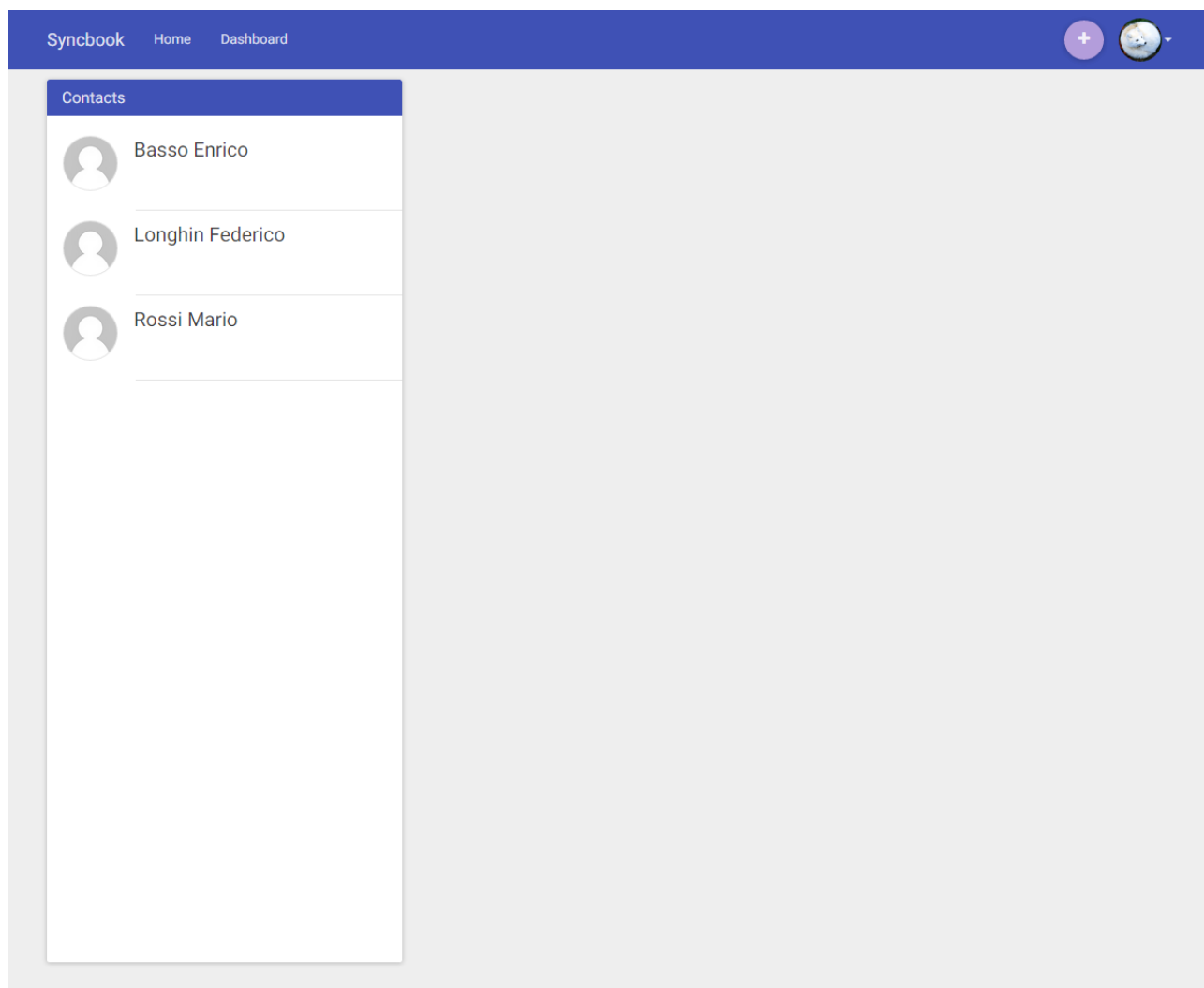
```
CREATE DATABASE IF NOT EXISTS `syncbook_users`;  
  
CREATE TABLE IF NOT EXISTS `syncbook_users`.`users` (  
  `user_id` INT(11) NOT NULL AUTO_INCREMENT  
  COMMENT 'auto incrementing user_id of each user, unique index',  
  `user_firstname` VARCHAR(64) COLLATE utf8_unicode_ci NOT NULL  
  COMMENT 'user's first name, unique',  
  `user_lastname` VARCHAR(64) COLLATE utf8_unicode_ci NOT NULL  
  COMMENT 'user's last name',  
  `user_name` VARCHAR(64) COLLATE utf8_unicode_ci NOT NULL  
  COMMENT 'user's name',  
  `user_password_hash` VARCHAR(255) COLLATE utf8_unicode_ci DEFAULT NULL  
  COMMENT 'user's password in salted and hashed format',  
  `user_email` VARCHAR(64) COLLATE utf8_unicode_ci NOT NULL  
  COMMENT 'user's email, unique',  
  `user_active` tinyint(1) NOT NULL DEFAULT '0'  
  COMMENT 'user's activation status',  
  `user_account_type` tinyint(1) NOT NULL DEFAULT '1'  
  COMMENT 'user's account type (basic, premium, etc)',  
  `user_has_avatar` tinyint(1) NOT NULL DEFAULT '0'  
  COMMENT '1 if user has a local avatar, 0 if not',  
  `user_remember_me_token` VARCHAR(64) COLLATE utf8_unicode_ci DEFAULT NULL  
  COMMENT 'user's remember-me cookie token',  
  `user_creation_timestamp` BIGINT(20) DEFAULT NULL  
  COMMENT 'timestamp of the creation of user's account',  
  `user_last_login_timestamp` BIGINT(20) DEFAULT NULL  
  COMMENT 'timestamp of user's last login',  
  `user_failed_logins` tinyint(1) NOT NULL DEFAULT '0'  
  COMMENT 'user's failed login attempts',  
  `user_last_failed_login` INT(10) DEFAULT NULL  
  COMMENT 'unix timestamp of last failed login attempt',  
  `user_activation_hash` VARCHAR(40) COLLATE utf8_unicode_ci DEFAULT NULL  
  COMMENT 'user's email verification hash string',  
  `user_password_reset_hash` CHAR(40) COLLATE utf8_unicode_ci DEFAULT NULL  
  COMMENT 'user's password reset code',  
  `user_password_reset_timestamp` BIGINT(20) DEFAULT NULL  
  COMMENT 'timestamp of the password reset request',  
  `user_provider_type` text COLLATE utf8_unicode_ci,  
  PRIMARY KEY (`user_id`),  
  UNIQUE KEY `user_name` (`user_name`),  
  UNIQUE KEY `user_email` (`user_email`)  
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci  
  COMMENT='user data';
```

### 6.1.3 Da HUGE a Syncbook

Per la realizzazione del progetto, HUGE è stato riadattato a quelle che sono le esigenze di Syncbook:

- la veste grafica è stata affidata a Bootstrap;
- la navigazione è stata organizzata in modo tale da avere una homepage (index) che fa da pagina introduttiva al progetto e una dashboard per il controllo delle funzionalità di Syncbook;
- sono state create due classi (ContactController e ContactModel) per la raccogliere i metodi atti alla manipolazione dei contatti;
- al momento della registrazione è stato reso obbligatorio l'inserimento di nome e cognome da parte dell'utente;
- eliminazione della tabella "notes" dal database.

### 6.1.3.1 Come si mostra l'applicazione



### 6.1.3.2 Inserimento di un contatto

Syncbook

Home

Dashboard

Contacts

Basso Enrico

Longhin Federico

Rossi Mario

Insert a new contact!

Prefix

Prefix

First Name

Federico

Middle Name

Middle Name

Last Name

Rossi

Suffix

Suffix

Birthday

mm/dd/yyyy

Phone

Home

+39 012 3456789

Mail

Home

Mail

Street Address

Home

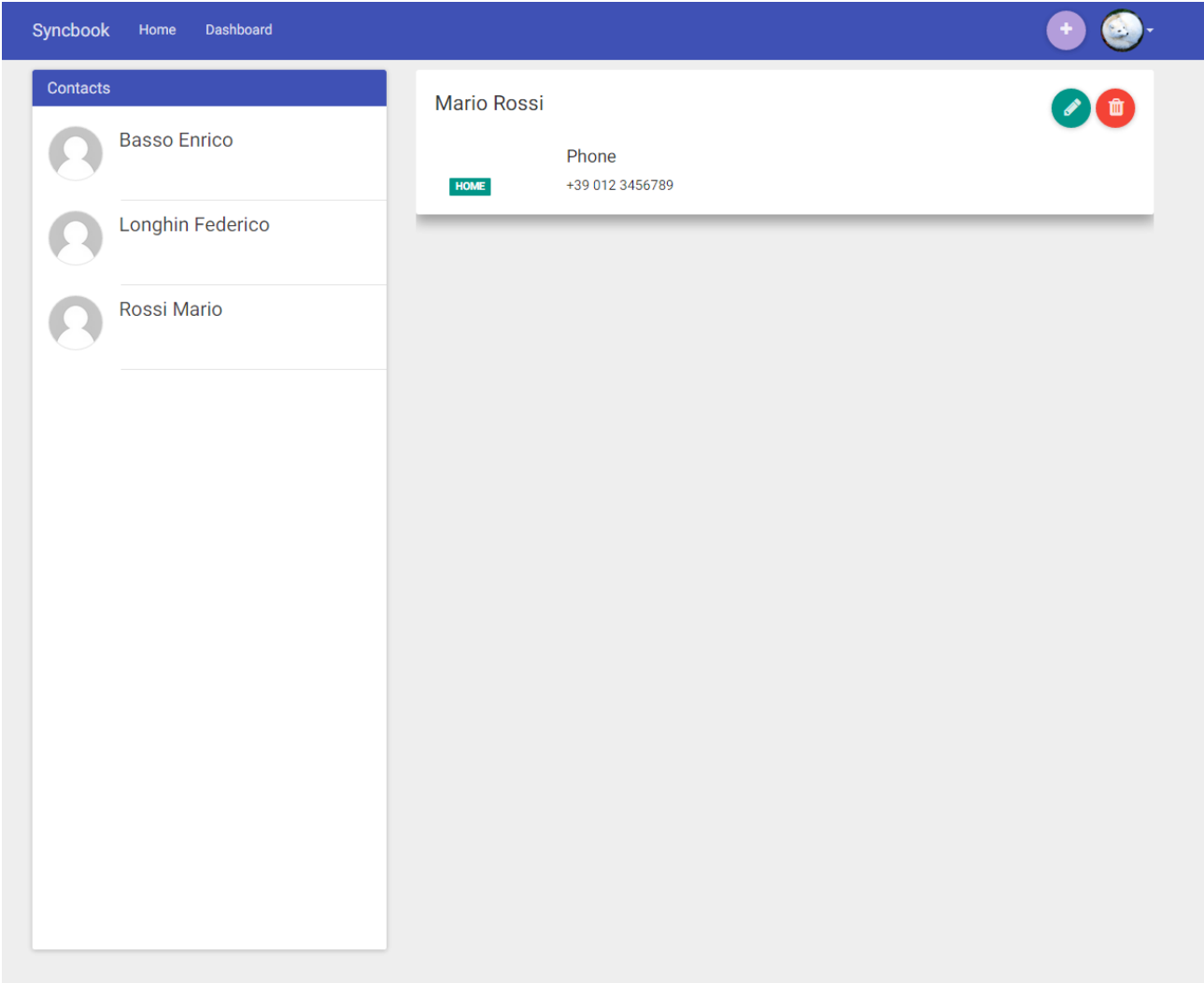
Street

City

Region

Postal code

6.1.3.3 Visualizzazione di un contatto





#### 6.1.3.4 Modifica di un contatto

Syncbook

Home

Dashboard

+

Contacts

Basso Enrico

Longhin Federico

Rossi Mario

Modify

Prefix

Prefix

First Name

Mario

Middle Name

Middle Name

Last Name

Rossi

Suffix

Suffix

Birthday

mm/dd/yyyy

Phone

Home

+39 012 3456789

Mail

Home

Mail

Street Address

Home

Street

City

Region

Postal code

13

## 6.2 Sabre.io

Framework PHP, creato con lo scopo di dare la possibilità ai programmatori di sviluppare applicazioni web completamente basate sui più recenti RFC di vCard e vCalendar nel modo più veloce e semplice possibile. Gestendo, inoltre, complessi protocolli come CardDAV, CalDAV e WebDAV attraverso il solo uso di istruzioni in linguaggio PHP.

Data la peculiare strutturazione della base di dati che viene utilizzata all'interno di Sabre/DAV è necessario creare un nuovo database per ogni utente che desidera creare un account all'interno di Syncbook.

Alla base di questo problema sta il fatto che, secondo i creatori del framework ogni utente dovrebbe essere gestito da un utente di livello più alto rispetto al proprio rendendo così più semplici, per gli amministratori d'azienda, le operazioni riguardanti la gestione dei server.

Se l'applicazione fosse gestita in questo modo sarebbe impossibile per i singoli utenti svolgere alcune fondamentali operazioni sulle loro rubriche, rendendo il servizio offerto, da alcuni punti di vista, peggiore rispetto a molte altre alternative che si possono trovare sul web.

Entra quindi in gioco il codice PHP che si può leggere sotto, utilizzato all'interno del file che si occupa della gestione degli accessi da dispositivi mobili, per reindirizzare l'utente all'interno della base di dati corretta durante il login.

```

<?php

// Controllo che il nome utente non sia già stato individuato in precedenza.
if (!isset($databaseUsername)) {
    // Viene ricavato l'URI utilizzato per accedere alla pagina Web corrente.
    $serverUri = $_SERVER["REQUEST_URI"];
    // Viene ricavato il path assoluto dello Script corrente all'interno del Server.
    $serverName = $_SERVER["SCRIPT_NAME"];

    /*
        Algoritmo utilizzato per ricavare l'username della persona che sta effettuando
        l'operazione di Login.
        Il nome utente viene poi salvato all'interno della variabile $databaseUsername.
        La variabile $databaseUsername sarà usata per connettere la persona al proprio database.

        Identificatore di un Database :
        sabredav_<username>
    */
    if (strpos($serverUri, '/principals/') !== false) {
        $databaseUsername = str_replace($serverName . "/principals/", "", $serverUri);
    } else if (strpos($serverUri, '/calendars/') !== false) {
        $databaseUsername = str_replace($serverName . "/calendars/", "", $serverUri);
    } else if (strpos($serverUri, '/addressbooks/') !== false) {
        $databaseUsername = str_replace($serverName . "/addressbooks/", "", $serverUri);
    } else {
        $databaseUsername = '';
    }

    if ($databaseUsername === '') {die();}

    // Necessario l'Username in minuscolo per la corretta connessione
    $databaseUsername = strtolower($databaseUsername);

    // Algoritmo di pulizia dell'Username da caratteri indesiderati immessi dalle applicazioni Client
    if (strpos($databaseUsername, "contacts") == false) {
        $databaseUsername = str_replace(' ', '-', $databaseUsername);
        $databaseUsername = preg_replace('/[^A-Za-z0-9\.-]/', '', $databaseUsername);
    } else {
        if (strpos($databaseUsername, "vcf") == false) {
            $databaseUsername = str_replace('/contacts/', '', $databaseUsername);
        } else {
            $databaseUsername = substr($databaseUsername, 0, strpos($databaseUsername, '/'));
        }
    }
}

?>

```

### 6.2.1 Creazione di una vCard

```
<?php

/**
 * Funzione utilizzata per creare una vCard dai dati forniti dall'utente durante il processo
 * di creazione di un contatto.
 *
 * @param $vCardObject
 * @return VObject\Component\VCard
 */
function mapperObjectCard($vCardObject) {

    // Mappatura del parametro UID.
    $vCard = new Sabre\VObject\Component\VCard([
        'UID' => ($vCardObject->UID !== "") ? $vCardObject->UID : Sabre\DAV\UUIDUtil::getUUID()
    ]);

    // Mappatura delle informazioni Standard.
    $contactDefault = $vCardObject->contactDefault;

    $string = NULL;

    // Serie di controlli per la presenza di nome/cognome e l'assenza "Prefix"/"Middle-Name"/"Suffix".
    if (!empty($contactDefault->contactPrefix)) {
        $string = $contactDefault->contactPrefix . " ";
    } else {$string = "";}

    if (!empty($contactDefault->contactMiddleName)) {
        $string = $string . $contactDefault->contactFirstName . " " . $contactDefault->contactMiddleName . " ";
    } else {$string = $string . $contactDefault->contactFirstName . " ";}

    if (!empty($contactDefault->contactSuffix)) {
        $string = $string . $contactDefault->contactLastName . " " . $contactDefault->contactSuffix;
    } else {$string = $string . $contactDefault->contactLastName;}

    // Operazioni di mappatura.
    $vCard->add('FN', $string);
    $vCard->add('N', [
        $contactDefault->contactLastName,
        $contactDefault->contactFirstName,
        $contactDefault->contactMiddleName,
        $contactDefault->contactPrefix,
        $contactDefault->contactSuffix
    ]);
}
```

```

// Mappatura della data di nascita.
$contactCompany = $vCardObject->contactCompany;

if ($contactCompany != NULL) {
    $dateTime = new \DateTime($contactCompany->contactBirthDate);
    $dateTime = $dateTime->format('Y-m-d\TH:i:s\Z');
    $vCard->add('BDAY', $dateTime);
}

// Mappatura dei dati relativi ai numeri di telefono.
$contactPhone = $vCardObject->contactPhone;

if ($contactPhone != NULL) {
    foreach($contactPhone as $phoneContainer) {
        $vCard->add('TEL', $phoneContainer->phoneValue, [
            'TYPE' => [
                $phoneContainer->phoneType,
                ($phoneContainer->phoneIsCell === "TRUE") ? 'CELL' : NULL,
                ($phoneContainer->phoneIsFax === "TRUE") ? 'FAX' : NULL,
                ($phoneContainer->phoneIsVoice === "TRUE") ? 'VOICE' : NULL
            ]
        ]);
    }
}

// Mappatura dei dati relativi agli indirizzi di posta elettronica.
$contactMail = $vCardObject->contactMail;

if ($contactMail != NULL) {
    foreach($contactMail as $mailContainer) {
        $vCard->add('EMAIL', $mailContainer->mailValue, [
            'TYPE' => [
                'INTERNET',
                $mailContainer->mailType
            ]
        ]);
    }
}

```

```

// Mappatura dei dati relativi agli indirizzi di abitazione/lavoro.
$contactAddress = $vCardObject->contactAddress;

if($contactAddress != NULL) {
    foreach($contactAddress as $addressContainer) {
        $vCard->add('ADR', [
            "",
            "",
            $addressContainer->addressStreet,
            $addressContainer->addressCity,
            $addressContainer->addressRegion,
            $addressContainer->addressPostalCode,
            $addressContainer->addressCountry,
        ], ['TYPE' => $addressContainer->addressType]);
    }
}

// Mappatura di dati relativi agli indirizzi web.
$contactInternet = $vCardObject->contactInternet;

if ($contactInternet != NULL) {
    foreach($contactInternet as $internetContainer) {
        $vCard->add('URL', $internetContainer->internetValue,
            ['TYPE' => $internetContainer->internetType]);
    }
}

// Mappatura dei dati relativi alle note.
if ($vCardObject->contactNotes != NULL) {
    $vCard->add('NOTE', $vCardObject->contactNotes);
}
return $vCard;
}

?>

```

## 6.2.2 Esempio di interfacciamento tra SabreDAV e HUGE

```
<?php

/**
 * Funzione utilizzata per interfacciare il sistema delle vCard con gli algoritmi peculiari di HUGE.
 * Algoritmo che ha lo scopo di ritornare una lista di Nomi-Cognomi di tutte le vCard di una certa rubrica.
 *
 * Struttura dell'Array di ritorno dalla funzione.
 * $returnArray = array(
 *   'UID' => array(
 *     'contactFirstName' => "",
 *     'contactLastName' => ""
 *   )
 * )
 *
 * @param \Sabre\CardDAV\AddressBook $addressBook
 * @param array $arrayUID
 * @return array|bool
 */
function vCardListRetrieve(\Sabre\CardDAV\AddressBook $addressBook, $arrayUID) {
    try {
        $returnArray = array();

        // Iterazione per tutti gli UID (attributo identificativo di una vCard) presenti all'interno del Database.
        foreach($arrayUID as $singleUID) {
            // Acquisizione dei dati presenti all'interno di una vCard, sotto forma di Object, attraverso l'UID.
            $vCardData = $addressBook->getChild($singleUID);
            // Acquisizione dei dati presenti all'interno di una vCard, sotto forma di vCard-Data.
            $vCardData = \Sabre\VObject\Reader::read($vCardData->get());
            // Utilizzo del mapper inverso a quello mostrato sopra con lo scopo di creare un Object
            // Partendo da un formato di tipo vCard-Data.
            $vCardObject = mapperCardObject($vCardData);

            // Popolazione dell'array di ritorno con i dati desiderati.
            $returnArray[$vCardObject->UID] = array(
                'contactFirstName' => $vCardObject->contactDefault->contactFirstName,
                'contactLastName' => $vCardObject->contactDefault->contactLastName
            );
        }

        return $returnArray;
    } catch (Exception $exceptionError) {
        error_log("Error in vCardListRetrieve Function - " . $exceptionError);
    }
    return FALSE;
}

?>
```

### 6.3 Bootstrap

Per quel che riguarda l'interfaccia grafica dell'applicazione è stato utilizzato Bootstrap, uno dei più famosi *front-end framework*. Al di là della sua fama, Bootstrap è stato scelto soprattutto per la sua vocazione *mobile first* e il numero di risorse di ogni genere disponibili online.

Le principali caratteristiche di questo framework sono il layout a griglie e la serie di classi CSS *preconfezionate* che possono essere implementate rapidamente nell'applicazione, dando fin da subito un risultato funzionale.

La struttura a colonne di Bootstrap permette di creare layout grafici in grado di adattarsi dinamicamente a qualunque display realizzando quello che viene definito *responsive design*.

Proprio per la sua natura esso è un framework da utilizzarsi solo nelle prime fasi di un progetto (da qui il nome), Bootstrap non fornisce un'ampia gamma di componenti aggiuntivi avendo uno stile grafico in sé povero. Per questo motivo si ha deciso di implementare una libreria CSS e JS in stile [material design](#): la quale, oltre a fornire una formattazione molto simile allo stile di Google, offre anche alcune componenti aggiuntive come le *floating labels* e finestre di dialogo.



### 6.3.1 Esempio di utilizzo di Bootstrap finestra di log-in

Una parte di codice HTML utilizzata per realizzare la finestra di login dell'applicazione.

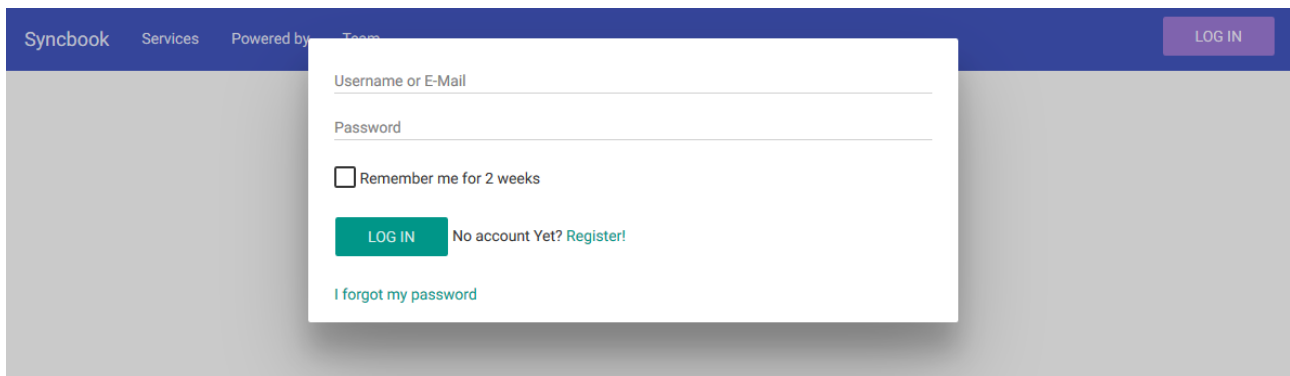
```
<div id="log-in-dialog" class="modal fade">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-body">
        <form
          class="form-horizontal"
          action="<?php echo Config::get('URL'); ?>login/login"
          method="post"
        >
          <fieldset>
            <div class="form-group">
              <div class="col-lg-12">
                <input
                  type="text"
                  class="form-control floating-label"
                  id="user_name" name="user_name"
                  placeholder="Username or E-Mail"
                  required
                />
              </div>
            </div>
            <div class="form-group">
              <div class="col-lg-12">
                <input
                  type="password"
                  class="form-control floating-label"
                  id="user_password"
                  name="user_password"
                  placeholder="Password"
                  required
                />
                <div class="checkbox">
                  <label>
                    <input
                      type="checkbox"
                      name="set_remember_me_cookie"
                    />
                    Remember me for 2 weeks
                  </label>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

```

        <div class="form-group">
            <div class="col-lg-12">
                <input
                    type="submit"
                    class="btn btn-primary"
                    value="Log in"
                />
                No account Yet?
                <a href="<?php
                    echo Config::get('URL');
                    ?>login/register">
                    Register!
                </a>
            </div>
        </div>
    </fieldset>
</form>
<div class="link-forgot-my-password">
    <a href="<?php
        echo Config::get('URL');
        ?>login/requestPasswordReset">
        I forgot my password
    </a>
</div>
</div>
</div>
</div>
</div>

```

#### 6.3.1.1 Il risultato



## 6.4 jQuery

jQuery è un framework JavaScript che ha come intento quello di snellire la programmazione semplificando la manipolazione degli elementi HTML e la gestione degli eventi del DOM (Document Object Model). Per un'applicazione che nasce in quello che viene definito il *web 2.0*, non è pensabile il non utilizzo di JS: sia dal punto di vista grafico, che dal punto di vista delle funzionalità.

Oltre alla semplificazione del codice, un'altra importante caratteristica di jQuery è che è sviluppato per funzionare allo stesso modo su più browser possibili, permettendo al programmatore di concentrarsi più sull'effettivo sviluppo dell'applicazione che sul funzionamento multi-piattaforma.

Se si vuole rendere un'applicazione più dinamica e veloce, dal punto di vista dell'utilizzo, è necessaria l'implementazione delle richieste asincrone. Questa tecnologia è implementata in jQuery con il nome AJAX e il metodo `$.ajax()` relativo. Questo metodo permette l'esecuzione di uno script nel server (nel nostro caso un controller/metodo specifico di HUGE) per poter fornire nuovi dati all'utente senza dover eseguire il refresh della pagina.

In Syncbook tutte le interazioni tra utente e database sono state eseguite attraverso chiamate AJAX.

Qui di seguito viene riportata la parte di codice che eseguita nel momento in cui l'utente salva delle modifiche a un contatto.

### 6.4.1 Esempio di utilizzo della funzione \$.ajax()

```
// Gestione dell'evento 'click' sul bottone 'Salva'.
$(document).on('click', '#btn_save_changes', function () {
    // Recupero dei dati dal form e costruzione dell'oggetto vCard.
    var vCard = getFormFields();
    vCard['UID'] = $(this).attr('data-uid');

    // Controllo sugli input.
    var control = inputControl(vCard);
    switch(control) {
        // Tutti i controlli sono stati rispettati, si procede con il salvataggio dei dati.
        case 0:
            // Preparazione ed esecuzione del metodo AJAX, vengono stabiliti:
            // - url del metodo da eseguire;
            // - parametri (data);
            // - metodo di passaggio dei parametri (POST);
            // - tipo di dati di ritorno (HTML);
            // - gestione degli errori;
            // - stampa a video di messaggi a operazione conclusa con successo.
            $.ajax({
                url : URL + 'contact/applychangestocontact',
                data : vCard,
                method : 'POST',
                dataType: 'html',
                error: function () {
                    // La richiesta non è andata a buon fine.
                    addNegativeFeedback("Internal error.");
                },
                success: function () {
                    $('#mainContainer').html("");
                    // Si mostra un feedback positivo che indica che l'operazione è andata a buon fine.
                    addSuccessFeedback("Contact modified.");

                    // Viene aggiornata la lista contatti.
                    loadContactList();
                }
            });
            break;

        case 1:
            // Non sono stati inseriti 'nome' e 'cognome'.
            alert("First name and last name fields are required!");
            break;

        case 2:
            // Tutti i campi 'address' devono essere riempiti.
            alert("Complete all the address fields!");
            break;
    }
});
```

## 6.5 RedBean PHP

Framework PHP utilizzato per implementare il paradigma di programmazione ORM (object-relational mapping) all'interno dei sorgenti di SabreDAV.

Si è presa la decisione di aggiungere questa funzionalità al progetto in modo tale da diminuire la quantità di codice scritto in fase di programmazione rendendo più semplici e lineari operazioni che sarebbero state molto complesse utilizzando mero linguaggio PHP.

### 6.5.1 ORM

L'Object-relation mapping è una tecnica di programmazione ampiamente utilizzata all'interno dei linguaggi di programmazione orientati all'utilizzo di strutture dati come gli oggetti. L'utilizzo di framework che implementano questo paradigma è spesso presente nel momento in cui è necessario interfacciare l'applicativo con uno specifico DBMS.

Nel caso del linguaggio PHP vengono implementati una serie di algoritmi con lo scopo di ampliare le funzionalità dell'estensione PDO rendendo più semplice per il programmatore la realizzazione di algoritmi che si appoggiano su Database altamente complessi e ramificati. Utilizzare l'ORM permette anche di risparmiare al programmatore molti controlli dovuti alla struttura di una base di dati.

#### 6.5.1.1 Tabella "persona"

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	<u>id</u>	int(10)		UNSIGNED	No	None	AUTO_INCREMENT
2	nome	varchar(128)	utf8mb4_unicode_ci		No	None	
3	cognome	varchar(128)	utf8mb4_unicode_ci		No	None	
4	data_nascita	date			No	None	

### 6.5.1.2 Interfacciamento attraverso ORM

```
<?php

try {
    // Connessione ad un Database di nome "example" situato in LocalHost.
    // Username: "root" - Password: "".
    R::setup('mysql:host=127.0.0.1;dbname=example', 'root', '');

    /*
        Non utilizzando il metodo sottostante RedBeanPHP utilizzerà gli schemi del Database in modo
        dinamico permettendo al programmatore di modificare la base di dati senza preoccuparsi delle
        possibili conseguenze nell'applicazione.

        Questa tecnica è molto utile in fase di sviluppo, in compenso le prestazioni diminuiscono,
        per questo motivo è quindi giusto utilizzare il metodo sottostante per bloccare gli schemi del
        Database permettendo le massime prestazioni.

        Tecnica che viene implementata solitamente alla pubblicazione dell'applicazione.
    */
    R::freeze(true);

    // Inizio di una transazione
    R::begin();

    // Acquisizione di un oggetto che presenta tutte le caratteristiche peculiari della tabella "persona"
    $beanPersona = R::dispense('persona');

    // Fase di popolzione dell'oggetto $beanPersona con i dati desiderati.
    // Nel caso in cui qualunque di questi dati non rispetti i vincoli descritti nella definizione
    // degli attributi, verrà generata un'eccezione. Opportunamente catturata attraverso
    // il costrutto "try-catch".
    $beanPersona->nome = "Pippo";
    $beanPersona->cognome = "Pluto";
    $beanPersona->dataNascita = 21/05/1953;

    // Creazione di un'istanza all'interno della tabella "persona", con i dati prima inseriti.
    // Il ritorno di questo metodo è il valore della chiave primaria che identifica la tupla.
    $beanPersonaID = R::store($beanPersona);

    // Chiusura della transazione precedentemente avviata in caso di successo dell'algoritmo.
    R::commit();

    // Chiusura della connessione con il Database
    R::close();
} catch (Exception $exceptionError) {
    // Chiusura della transazione precedentemente avviata in caso di insuccesso dell'algoritmo.
    R::rollback();

    // Chiusura dello script corrente
    die("Error: $exceptionError");
}

?>
```

## 6.6 Git

Sistema software di controllo di versione distribuito, creato da Linus Torvalds nel 2005.

Diventata il leader nella sua categoria con il passare del tempo, questa piattaforma ha fatto della velocità e dell'integrità dei dati i suoi punti di forza.

Ogni directory di lavoro creata attraverso Git identifica un repository cioè una struttura dati che contiene dei metadati che permettono di tracciare tutte le azioni che sono compiute in fase di progettazione.

### 6.6.1 Commit

All'interno di un repository ogni modifica al codice sorgente è possibile soltanto attraverso un'operazione di commit: un insieme di metadati a cui è possibile dare una descrizione. Attraverso questa metodologia, ogni membro del repository ha la possibilità di conoscere le differenze che sono state apportate all'interno dei file a cui un commit fa riferimento.

### 6.6.2 Branch

Attraverso una particolare tipologia di commit è possibile creare un branch all'interno di un repository, differente dal *master branch* che identifica la root directory e che viene creato attraverso il processo di inizializzazione di un repository.

Un branch è un insieme di metadati che permette l'identificazione di un insieme di file/cartelle all'interno di un repository. I branch vengono, molto spesso, utilizzati per differenziare le varie aree di lavoro dei membri del repo o, più semplicemente, per separare vari componenti di un progetto non collegati tra loro.

### 6.6.3 GitHub

Dato il fatto che Git nasce, principalmente, come una piattaforma per lavorare in gruppi di progetto composti da più persone, che molte volte vivono in luoghi distanti tra loro, è necessaria una piattaforma Server dove tutti possano accedere per apportare delle modifiche al Repository.

Viene fondato, quindi, nel 2008 GitHub. Sito web pubblico con lo scopo di fornire una completa piattaforma di hosting per i propri Repository. Esistono due versioni principali di questo servizio:

- Gratuito, per chiunque desideri pubblicare il proprio lavoro con il resto del mondo, avendo quindi la possibilità di essere aiutati in caso di bisogno.
- A pagamento, per coloro che vogliono che i propri repository rimangano privati e che solo alcune persone possano accedere ad essi.

Queste due versioni presentano uguali capacità di gestione del codice, complete di statistiche e informazioni in tempo reale riguardanti il proprio lavoro.

### 6.6.3.1 Repository “Home”

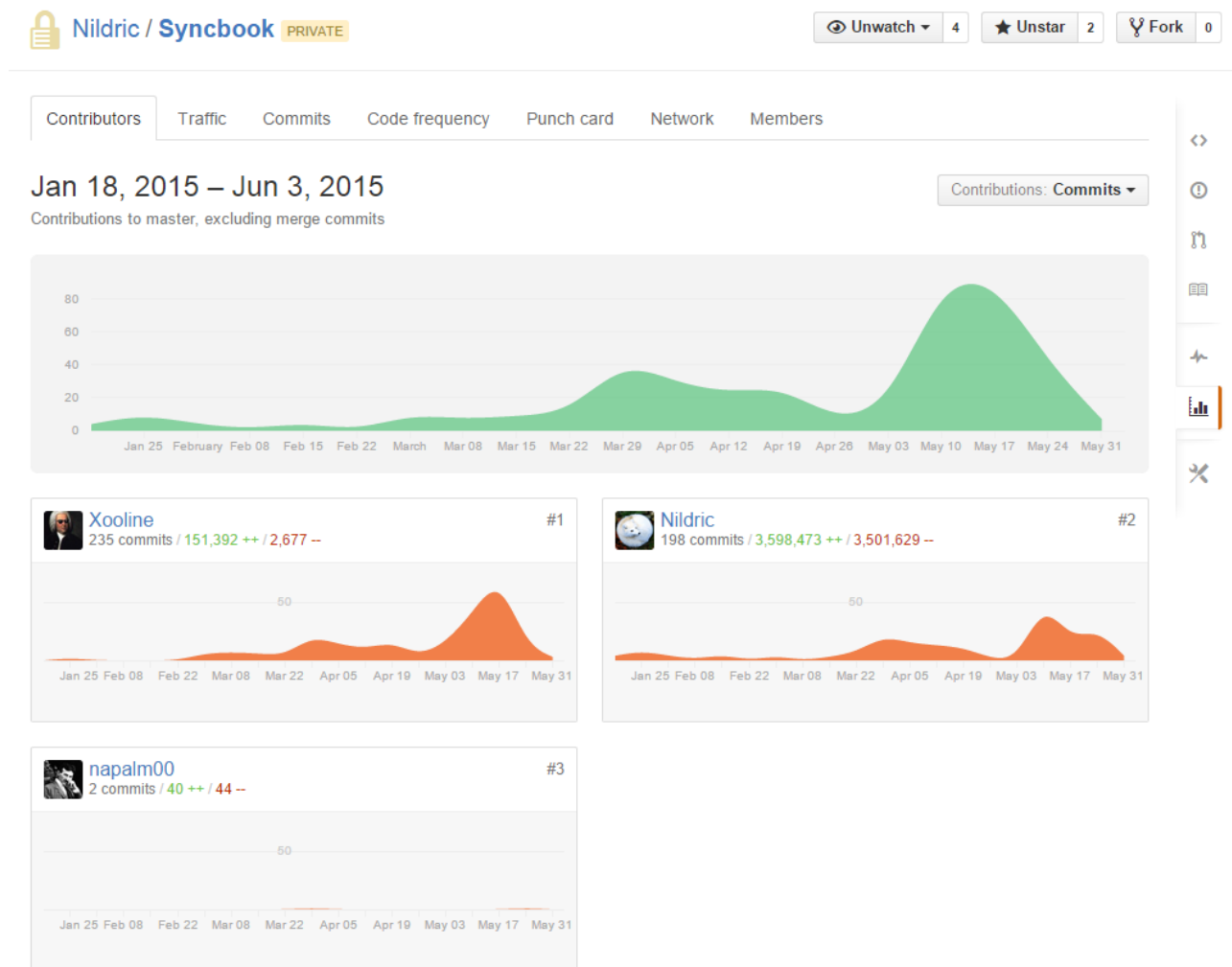
The screenshot shows the GitHub repository page for 'Nildric / Syncbook'. At the top, the repository name is followed by 'PRIVATE' and statistics: 451 commits, 2 branches, 1 release, and 3 contributors. Below this is a table of recent commits, including updates to README.md, .idea, cfg, lib, sql, src, test, .gitignore, .travis.yml, and README.md. On the right sidebar, there are links to Code, Issues (2), Pull requests (0), Wiki, Pulse, Graphs, and Settings. At the bottom of the sidebar, there are options to clone the repository via SSH, HTTPS, or Subversion, and buttons for 'Clone in Desktop' and 'Download ZIP'. The main content area displays the README.md file, which features the 'Syncbook' logo and a purple circular icon with a stylized 'S'.

L'immagine mostra la struttura della pagina iniziale di un repository all'interno di GitHub, vengono presentate le seguenti funzioni:

- fondatore del Repository e nome dello stesso (Nildric/Syncbook);
- numero di commit attualmente presenti all'interno del repository (451 commits);
- numero di branches attualmente presenti all'interno del repository (2 branches);
- numero di release attualmente presenti all'interno del repository (1 release);
- numero di contributors attualmente presenti all'interno del repository (2 contributors);
- struttura della Root directory del repository;
- numero di issues attualmente presenti all'interno del repository (2 issues);
- numero di pull requests attualmente presenti all'interno del repository (2 pull requests);
- link alla wiki del repository;
- link ai grafici del repository, descritti nella sezione sottostante.




### 6.6.3.2 Repository “Graphs”






L'immagine mostra il portale attraverso il quale è possibile vedere:

- la data del primo commit del repository e la data dell'ultimo (Jan 18, 2015 - Jun 3, 2015);
- un grafico riassuntivo della frequenza dei commit compiuti durante il periodo sopra citato;
- la frequenza dei commit di ogni membro del repository nel tempo;
- il numero di righe di codice aggiunte e rimosse da ogni membro nel tempo.































### 6.6.3.3 Repository “Commits”

 Nildric / Syncbook PRIVATE













 Unwatch 4  Unstar 2  Fork 0

branch: master

Commits on May 27, 2015

	<b>Deleted useless paragraph.</b> Xooline authored 7 days ago	 2cfc22a	
	<b>Added "internet" and "notes" fields.</b> Xooline authored 7 days ago	 0b5c73c	
	<b>Added "internet" and "notes" fields.</b> Xooline authored 7 days ago	 88b223f	
	<b>New profile interface.</b> Xooline authored 7 days ago	 c2e9e0a	
	<b>Changed &lt;section&gt;&lt;row&gt; to &lt;row&gt;&lt;section&gt;</b> Nildric authored 7 days ago	 16446e1	
	<b>Minor changes to scrolling-nav.css</b> Nildric authored 7 days ago	 28b579b	
	<b>Changes to base.css</b> Nildric authored 7 days ago	 5918b78	
	<b>Updated images in homepage</b> Nildric authored 7 days ago	 6d1fc1a	
	<b>Added favicon to Website</b> Nildric authored 7 days ago	 3b6c593	
	<b>Updated all the images used in Homepage</b> Nildric authored 7 days ago	 c591a76	

Commits on May 25, 2015

	<b>Added a way to handle the "FN" property even if some values that are ...</b> ... Nildric authored 9 days ago	 07a070c	
	<b>Changed the way sections works on the NavBar.</b> Nildric authored 9 days ago	 eb935ab	
	<b>Changed saveNewEmailAddress() Function to handle SabreDAV Database E-...</b> ... Nildric authored 9 days ago	 6012f6a	
	<b>Added "InternetType" and "InternetValue" Values into ContactModel.</b> ... Nildric authored 9 days ago	 6536c1b	

L'immagine mostra una parte della lista di Commit compiuti durante il ciclo di vita del Repository.

Per ogni Commit è possibile vedere:

- l'autore (Xooline/Nildric);
- la data in cui è stato fatto (May 27, 2015);
- la descrizione scritta dall'autore a riguardo;
- quanto tempo è trascorso tra la data di creazione e la data corrente (7 days ago);
- il codice identificativo (2cfc22a).

### 6.6.3.4 Esempio di "Commit"

Nildric / Syncbook PRIVATE

Unwatch 4 Unstar 2 Fork 0

New profile interface. master Browse files

Xooline authored 7 days ago 1 parent 16446e1 commit c2e9e0ab55878171763dc91ed328c815952f53

Showing 1 changed file with 17 additions and 12 deletions. Unified Split

src/huge/application/view/login/showProfile.php View

```
@@ -1,20 +1,25 @@
1 <div class="container #ix-navbar">
2 <!-- echo out the system feedback (error and success messages) -->
3 </php $this->renderFeedbackMessages(); >
4 -
5 - <div class="col-sm-8">
6 - <div class="panel-warning">
7 - <div class="panel-heading"><? $this->user_name; ></div>
8 - <div class="panel-body">
9 - <div class="col-sm-6">Your email:</div>
10 - <div class="col-sm-6"><? $this->user_email; ></div>
11 - <div class="col-sm-6">Your avatar image (on gravatar.com)</div>
12 - <div class="col-sm-6">
13 - 
14 -
15 </div>
16
17 -
18 </div>
19 </div>
20 </div>

1 <div class="container #ix-navbar">
2 <!-- echo out the system feedback (error and success messages) -->
3 </php $this->renderFeedbackMessages(); >
4 + <div class="col-sm-4"></div>
5 + <div class="col-sm-4">
6 + <div class="well no-padding" style="padding: 0px;">
7 + <div class="card hovercard" style="display: block;">
8 + <div class="cardheader">
9
10 + </div>
11 + <div class="avatar">
12 + 
13 + </div>
14 + <div class="info">
15 + <div class="title">
16 + <a target="_blank" href="https://twitter.com/LonginFederico"><? $this->user_name; ></a>
17 + </div>
18 + <div class="desc"><? $this->user_email; ></div>
19 + <div class="desc">Profile picture powered by <img alt="Gravatar logo" /></div>
20 + </div>
21 + <div class="bottom"></div>
22 </div>
23 </div>
24 + <div class="col-sm-4"></div>
25 </div>
```

In questa immagine vengono mostrate le modifiche al File a cui uno specifico Commit fa riferimento.

## 6.7 Server

Tutti i servizi di Syncbook sono ospitati in un server cloud su [digitalocean.com](https://digitalocean.com).

Inoltre è stato registrato il dominio *syncbook.me* configurando i DNS della macchina virtuale, in aggiunta è stata installata una certificazione SSL, ormai diventata indispensabile per le applicazioni web.

Il server si basa su Ubuntu 14.04 LTS, per i servizi HTTP è stato installato Apache 2, PHP e MySQL. Per sfruttare appieno il dominio, e per fornire la funzione di registrazione agli utenti di Syncbook, è stato installato un mail server, nel particolare Postfix. La configurazione del mail server ha conseguito il seguente risultato su [mail-tester.com](https://mail-tester.com):



## 7 Conclusioni

Allo stato attuale del progetto possiamo dire di aver raggiunto in modo soddisfacente gli obiettivi prefissati: Syncbook permette di organizzare una rubrica di contatti rispettando le regole dettate dagli standard vCard e mette a disposizione i dati inseriti dall'utente anche su altri dispositivi sincronizzati tramite WebDAV. Allo stesso tempo non sono state ancora implementate delle funzioni che potrebbero allargare l'utilizzo di Syncbook anche a gruppi di lavoro, oltre che al singolo utente: ad esempio la possibilità di creare più rubriche e di condividerle con delle persone selezionate. In ogni caso, grazie ai vari framework utilizzati, il progetto è stato sviluppato in modo tale da poter accogliere nuove modifiche senza dover stravolgere tutto il lavoro fatto in precedenza.

Per la realizzazione del progetto, abbiamo avuto modo di conoscere ed utilizzare strumenti che stanno alla base dello sviluppo di applicazioni, in questo caso web, all'interno del mondo del lavoro. Lo studio di questi applicativi, svolto da autodidatti ci ha permesso di sviluppare l'applicazione obiettivo del progetto in modo professionale e spendibile nel mondo reale.

### 7.1 Problematiche

L'unica problematica degna di nota, riguarda la creazione di un Database inattivo fino alla conferma dell'indirizzo e-mail da parte dell'utente. Visto che una base di dati viene creata in corrispondenza con la registrazione di un nuovo utente, voleva essere trovata una modalità attraverso la quale l'utente non potesse utilizzare le credenziali immesse durante la registrazione, dal dispositivo mobile, senza aver confermato il proprio indirizzo e-mail.

```
-- Creazione di un Database inattivo, non gestito dagli algoritmi
dell'applicazione.
CREATE DATABASE IF NOT EXISTS sabredav_<username>_unactive;

-- Creazione delle varie tabelle del Database.

-- Nel caso in cui l'indirizzo mail venga confermato dall'utente, viene
-- eseguito la seguente serie di istruzioni.
-- Creazione di un Database gestito dall'applicazione
CREATE DATABASE IF NOT EXISTS sabredav_<username>;

-- Spostamento delle tabelle e del loro contenuto dal Database inattivo a
-- quello attivo.
RENAME TABLE sabredav_<username>_unactive.addressbookchanges TO
sabredav_<username>.addressbookchanges;
RENAME TABLE sabredav_<username>_unactive.addressbooks TO
sabredav_<username>.addressbooks;
RENAME TABLE sabredav_<username>_unactive.cards TO sabredav_<username>.cards;
RENAME TABLE sabredav_<username>_unactive.groupmembers TO
sabredav_<username>.groupmembers;
RENAME TABLE sabredav_<username>_unactive.locks TO sabredav_<username>.locks;
RENAME TABLE sabredav_<username>_unactive.principals TO
sabredav_<username>.principals;
RENAME TABLE sabredav_<username>_unactive.propertystorage TO
sabredav_<username>.propertystorage;
RENAME TABLE sabredav_<username>_unactive.users TO sabredav_<username>.users;

-- Cancellazione del Database inattivo.
DROP DATABASE sabredav_<username>_unactive;
```

## 7.2 Implementazioni future

### 7.2.1 Gestione di più rubriche

Gestire più di una rubrica rispetto a quella offerta di default dalla piattaforma potrebbe essere, per molte persone, una funzionalità importante per poter gestire al meglio la propria lista di contatti, dividendola in categorie. Questa tipologia di gestione è supportata dal protocollo CardDAV ma non è stata implementata perché con la differente gestione della base di dati, necessaria per il corretto funzionamento di Syncbook, SabreDAV non permette di creare, modificare e cancellare più rubriche.

### 7.2.2 Importazione ed esportazione di vCard

Durante l'utilizzo della piattaforma potrebbe essere necessaria, per l'utente, la capacità di importare ed esportare i propri contatti da/a dispositivi che non utilizzano correttamente la formattazione standard.

Si è deciso di non implementare questa funzionalità principalmente per due motivi:

- 1) Il fattore di esportazione sta alla base della finalità di questo progetto, configurando il proprio dispositivo per poter utilizzare il protocollo CardDAV riuscendo ad avere contatti standardizzati ovunque si desideri. Per quanto riguarda l'esportazione della vCard sotto la forma di file questa funzione non è stata implementata a causa di problemi dovuti dalle modifiche fatte alla base di dati utilizzata da SabreDAV.
- 2) Il fattore importazione, data la politica delle grandi aziende a riguardo l'uso degli RFC, sarebbe stato molto complesso da implementare perché avere un algoritmo di mappatura per ogni tipologia di servizio offerto per la gestione dei propri contatti è quasi impossibile. Un team di programmatori avrebbe dovuto essere incaricato solo di occuparsi delle varie sfaccettature di questo problema.

## 8 Bibliografia

- [Articolo](#) sulle problematiche riguardanti l'interoperabilità del formato vCard.

### 8.1 Documentazione tecnica

- [HUGE](#);
- [Sabre.io](#);
- [Bootstrap](#);
- [Material Design](#)
- [jQuery](#);
- [RedBean PHP](#);
- [Git](#);
- [GitHub](#).

#### 8.1.1 RFC

- [6350](#) vCard;
- [6352](#) WebDAV/CardDAV.

### 8.2 Siti di riferimento

- [DigitalOcean](#);
- [DNSimple](#);
- [Namecheap](#);
- [StackOverflow](#);
- [BootSnip](#).

## 9 Si ringraziano

- Callegari Filippo, per il lavoro svolto nella gestione della parte server del progetto;
- Sponchiado Francesco, per il lavoro svolto durante alcune fasi dello sviluppo di Syncbook.