

# **Implementazione e Test creazione automatica dei test**

**Titolo del progetto:** Creazione Automatica dei Test  
**Alunni:** Alessandro Narciso & Dragan Jerkic  
**Classe:** I3AA  
**Anno scolastico:** 2016/2017  
**Docente responsabile:** Luca Muggiasca

1	Implementazione .....	3
1.1	Config.....	3
1.2	Login .....	4
1.3	Registrazione .....	8
1.4	WEB.....	11
1.5	PDF.....	13
2	Test .....	15
2.1	Protocollo di test.....	15
2.2	Risultati test .....	22
3	Consuntivi .....	23
3.1	Gantt consuntivo .....	23
3.2	Costo Consuntivo .....	24
4	Conclusioni.....	25
4.1	Sviluppi futuri .....	25
4.2	Considerazioni personali .....	25
5	Sitografia .....	25
6	Allegati .....	25

## 1 Implementazione

### 1.1 Config

In questa parte di codice ci occupiamo di controllare la connessione con il database, con la funzione `databaseConnection()`.

```
class Config {
    /**
     * @var object $db_connection The database connection
     */
    protected $db_connection = null;

    /**
     * Checks if database connection is opened. If not, then this method tries to open it.
     * @return bool Success status of the database connecting process
     */
    protected function databaseConnection(){
        // if connection already exists
        if ($this->db_connection != null) {
            return true;
        } else {
            try {
                $this->db_connection = new PDO('mysql:host='. DB_HOST .';dbname='. DB_NAME .';charset=utf8', DB_USER, DB_PASS);
                return true;
            } catch (PDOException $e) {
                $this->errors[] = MESSAGE_DATABASE_ERROR . $e->getMessage();
            }
        }
        return false;
    }
}
```

Questo codice è la sezione (nel `config.php`) dove definiamo delle costanti per la connessione al database.

```
/**
 * Configuration for: Database Connection
 * This is the place where your database login constants are saved
 *
 * For more info about constants please @see http://php.net/manual/en/function.define.php
 * If you want to know why we use "define" instead of "const" @see http://stackoverflow.com/q/2447791/1114320
 *
 * DB_HOST: database host, usually it's "127.0.0.1" or "localhost", some servers also need port info
 * DB_NAME: name of the database. please note: database and database table are not the same thing
 * DB_USER: user for your database. the user needs to have rights for SELECT, UPDATE, DELETE and INSERT.
 *          by the way, it's bad style to use "root", but for development it will work.
 * DB_PASS: the password of the above user
 */
define("DB_HOST", "127.0.0.1");
define("DB_NAME", "cat");
define("DB_USER", "root");
define("DB_PASS", "root");
```

## 1.2 Login

In questo pezzo di codice possiamo vedere il costruttore della classe di login. Questo costruttore viene passato ogni volta che la pagina viene ricaricata in modo da controllare in qualsiasi momento una possibile richiesta da parte dell'utente.

```

/**
 * the function "__construct()" automatically starts whenever an object of this class is created,
 * you know, when you do "$login = new Login();"
 */
public function __construct(){
    // create/read session
    session_start();

    // check the possible login actions:
    // 1. logout (happen when user clicks logout button)
    // 2. login via session data (happens each time user opens a page on your php project AFTER he has successfully logged in via the login form)
    // 3. login via cookie
    // 4. login via post data, which means simply logging in via the login form. after the user has submit his login/password successfully, his
    //    logged-in-status is written into his session data on the server. this is the typical behaviour of common login scripts.

    // if user tried to log out
    if (isset($_GET["logout"])) {
        $this->doLogout();

        // if user has an active session on the server
    } elseif (!empty($_SESSION['user_name']) && ($_SESSION['user_logged_in'] == 1)) {
        $this->loginWithSessionData();

        // checking for form submit from editing screen
        // user try to change his username
        if (isset($_POST["user_edit_submit_email"])) {
            // function below uses use $_SESSION['user_id'] et $_SESSION['user_email']
            $this->editUserEmail($_POST['user_email']);
            // user try to change his password
        } elseif (isset($_POST["user_edit_submit_password"])) {
            // function below uses $_SESSION['user_name'] and $_SESSION['user_id']
            $this->editUserPassword($_POST['user_password_old'], $_POST['user_password_new'], $_POST['user_password_repeat']);
        }

        // login with cookie
    } elseif (isset($_COOKIE['rememberme'])) {
        $this->loginWithCookieData();

        // if user just submitted a login form
    } elseif (isset($_POST["login"])) {
        if (isset($_POST['user_rememberme'])) {
            $_POST['user_rememberme'] = null;
        }
        $this->loginWithPostData($_POST['user_email'], $_POST['user_password'], $_POST['user_rememberme']);
    }
}

```

Questo è il metodo principale del login che viene chiamato quando l'utente fa la richiesta dalla pagina di login.

```
/**
 * Logs in with the data provided in $_POST, coming from the login form
 */
private function loginWithPostData($user_email, $user_password, $user_rememberme){
    if (empty($user_email)) {
        $this->errors[] = MESSAGE_USERNAME_EMPTY;
    } else if (empty($user_password)) {
        $this->errors[] = MESSAGE_PASSWORD_EMPTY;

        // if POST data (from login form) contains non-empty user_name and non-empty user_password
    } else {
        // user can login with his username or his email address.
        // if user has not typed a valid email address, we try to identify him with his user_name
        if (!filter_var($user_email, FILTER_VALIDATE_EMAIL)) {
            $this->errors[] = MESSAGE_EMAIL_INVALID;

            // if user has typed a valid email address, we try to identify him with his user_email
        } else if ($this->databaseConnection()) {
            // database query, getting all the info of the selected user
            $query_user = $this->db_connection->prepare('SELECT * FROM users WHERE user_email = :user_email');
            $query_user->bindValue(':user_email', trim($user_email), PDO::PARAM_STR);
            $query_user->execute();
            // get result row (as an object)
            $result_row = $query_user->fetchObject();
        }

        // if this user not exists
        if (!isset($result_row->user_id)) {
            // was MESSAGE_USER_DOES_NOT_EXIST before, but has changed to MESSAGE_LOGIN_FAILED
            // to prevent potential attackers showing if the user exists
            $this->errors[] = MESSAGE_LOGIN_FAILED;
        } else if ($result_row->user_ban) {
            // User banned
            $this->errors[] = MESSAGE_BAN.$result_row->user_ban_text;
        } else if (($result_row->user_failed_logins >= 5) && ($result_row->user_last_failed_login > (time() - 30))) {
            $this->errors[] = MESSAGE_PASSWORD_WRONG_TIMES;
            // using PHP 5.5's password_verify() function to check if the provided passwords fits to the hash of that user's password
        } else if (!password_verify($user_password, $result_row->user_password_hash)) {
            // increment the failed login counter for that user
            $sth = $this->db_connection->prepare('UPDATE users '
                . 'SET user_failed_logins = user_failed_logins+1, user_last_failed_login = :user_last_failed_login '
                . 'WHERE user_email = :user_email');
            $sth->execute(array(':user_email' => $user_email, ':user_last_failed_login' => time()));

            $this->errors[] = MESSAGE_PASSWORD_WRONG;
        }
    }
}
```

Questo è il codice HTML della pagina di login.

```
<form method="post" class="form-horizontal m-t-20">
    <div class="form-group">
        <input class="form-control" type="text" name="user_email" placeholder="Email" value="{<?php echo EMAIL_TEMP; ?>" required>
    </div>
    <div class="form-group">
        <input class="form-control" type="password" name="user_password" placeholder="Password" required>
    </div>
    <div class="form-group">
        <div class="checkbox checkbox-primary">
            <input id="checkbox-signup" type="checkbox" name="user_rememberme">
            <label for="checkbox-signup">Ricordami</label>
        </div>
    </div>
    <div class="form-group text-center m-t-40">
        <button class="btn btn-success btn-block waves-effect waves-light" name="login" type="submit">Accedi</button>
    </div>
</form>
```

Qui settiamo i dati presi dal database nella sessione e nelle variabili di classe che poi potranno essere riutilizzati dalla sessione per far accedere l'utente.

```
else {
    // write user data into PHP SESSION [a file on your server]
    $_SESSION['user_id'] = $result_row->user_id;
    $_SESSION['user_name'] = $result_row->user_name;
    $_SESSION['user_surname'] = $result_row->user_surname;
    $_SESSION['user_email'] = $result_row->user_email;
    $_SESSION['user_admin'] = $result_row->user_admin;
    $_SESSION['user_logged_in'] = 1;

    // declare user id, set the login status to true
    $this->user_id = $result_row->user_id;
    $this->user_name = $result_row->user_name;
    $this->user_surname = $result_row->user_surname;
    $this->user_email = $result_row->user_email;
    $this->user_admin = $result_row->user_admin;
    $this->user_is_logged_in = true;

    // reset the failed login counter for that user
    $sth = $this->db_connection->prepare('UPDATE users '
    . 'SET user_failed_logins = 0, user_last_failed_login = NULL '
    . 'WHERE user_id = :user_id AND user_failed_logins != 0');
    $sth->execute(array(':user_id' => $result_row->user_id));

    // if user has checked the "remember me" checkbox, then generate token and write cookie
    if (isset($user_rememberme)) {
        $this->newRememberMeCookie();
    } else {
        // Reset remember-me token
        $this->deleteRememberMeCookie();
    }
}
```

Questa è la funzione che userà l'utente per eseguire la disconnessione.

```
/**
 * Perform the logout, resetting the session
 */
public function doLogout(){
    $this->deleteRememberMeCookie();

    $_SESSION = array();
    session_destroy();

    $this->user_is_logged_in = false;
    $this->messages[] = MESSAGE_LOGGED_OUT;
}
```



Questa è la libreria usata per codificare la password come hash per trasferirla sul database in modo sicuro.

```
<?php
/**
 * A Compatibility library with PHP 5.5's simplified password hashing API.
 *
 * @author Anthony Ferrara <ircmaxell@php.net>
 * @license http://www.opensource.org/licenses/mit-license.html MIT License
 * @copyright 2012 The Authors
 */

if (!defined('PASSWORD_DEFAULT')) {

    define('PASSWORD_BCRYPT', 1);
    define('PASSWORD_DEFAULT', PASSWORD_BCRYPT);

    /**
     * Hash the password using the specified algorithm
     *
     * @param string $password The password to hash
     * @param int $algo The algorithm to use (Defined by PASSWORD_* constants)
     * @param array $options The options for the algorithm to use
     *
     * @return string|false The hashed password, or false on error.
     */
    function password_hash($password, $algo, array $options = array()) { ...
    }

    /**
     * Determine if the password hash needs to be rehashed according to the options provided
     *
     * If the answer is true, after validating the password using password_verify, rehash it.
     *
     * @param string $hash The hash to test
     * @param int $algo The algorithm used for new password hashes
     * @param array $options The options array passed to password_hash
     *
     * @return boolean True if the password needs to be rehashed.
     */
    function password_needs_rehash($hash, $algo, array $options = array()) { ...
    }

    /**
     * Verify a password against a hash using a timing attack resistant approach
     *
     * @param string $password The password to verify
     * @param string $hash The hash to verify against
     *
     * @return boolean If the password matches the hash
     */
    function password_verify($password, $hash) { ...
    }
}
```

### 1.3 Registrazione

Con questo pezzo di codice abbiamo creato il form per la parte di registrazione, qui ci sono tutti i campi che usciranno nella pagina web:

```
<form method="post" class="form-horizontal m-t-20">
  <div class="form-group">
    <div class="col-xs-12">
      <input class="form-control" type="text" name="user_name" placeholder="Nome" required>
    </div>
  </div>
  <div class="form-group">
    <div class="col-xs-12">
      <input class="form-control" type="text" name="user_surname" placeholder="Cognome" required>
    </div>
  </div>
  <div class="form-group">
    <div class="col-xs-12">
      <input class="form-control" type="email" name="user_email" placeholder="Email" required>
    </div>
  </div>
  <div class="form-group">
    <div class="col-xs-12">
      <input class="form-control" type="password" name="user_password_new" placeholder="Password" required>
    </div>
  </div>
  <div class="form-group">
    <div class="col-xs-12">
      <input class="form-control" type="password" name="user_password_repeat" placeholder="Conferma password" required>
    </div>
  </div>
  <div class="form-group">
    <div class="col-xs-12">
      <div class="checkbox checkbox-primary">
        <input id="checkbox-signup" type="checkbox" required>
        <label for="checkbox-signup">Accetto i <a href="#">Termini e le Condizioni</a></label>
      </div>
    </div>
  </div>
  <div class="form-group text-center m-t-40">
    <div class="col-xs-12">
      <button class="btn btn-success btn-block text-uppercase waves-effect waves-light" name="register" type="submit">
        Registrati
      </button>
    </div>
  </div>
</form>
```

In questa parte di codice abbiamo implementato il costruttore della classe register. Qui vediamo se la sessione è aperta e poi controlliamo se la registrazione ha un valore e dopo registriamo l'utente con i valori inseriti precedentemente.

```
/**
 * the function "__construct()" automatically starts whenever an object of this class is created,
 * you know, when you do "$login = new Login();"
 */
public function __construct()
{
    session_start();

    // if we have such a POST request, call the registerNewUser() method
    if (isset($_POST["register"])) {
        $this->registerNewUser($_POST['user_name'], $_POST['user_surname'], $_POST['user_email'], $_POST['user_password_new'], $_POST['user_password_repeat']);
    }
    // if we have such a GET request, call the verifyNewUser() method
}
}
```



## Implementazione e Test creazione automatica dei test

Questa è la parte dove registriamo l'utente, e qui vengono fatti tutti i vari controlli per il nome, per il cognome, per l'email, per la password e per la verifica della password:

```
/**
 * handles the entire registration process. checks all error possibilities, and creates a new user in the database if
 * everything is fine
 */
private function registerNewUser($user_name, $user_surname, $user_email, $user_password, $user_password_repeat)
{
    // we just remove extra space on username and email
    $user_name = trim($user_name);
    $user_surname = trim($user_surname);
    $user_email = trim($user_email);

    // check provided data validity
    // TODO: check for "return true" case early, so put this first
    if (empty($user_name)) {
        $this->errors[] = MESSAGE_NAME_EMPTY;
    }
    elseif (empty($user_surname)) {
        $this->errors[] = MESSAGE_SURNAME_EMPTY;
    }
    elseif (empty($user_password) || empty($user_password_repeat)) {
        $this->errors[] = MESSAGE_PASSWORD_EMPTY;
    }
    elseif ($user_password !== $user_password_repeat) {
        $this->errors[] = MESSAGE_PASSWORD_BAD_CONFIRM;
    }
    elseif (strlen($user_password) < 6) {
        $this->errors[] = MESSAGE_PASSWORD_TOO_SHORT;
    }
    elseif (strlen($user_name) > 64 || strlen($user_name) < 2) {
        $this->errors[] = MESSAGE_NAME_BAD_LENGTH;
    }
    elseif (!preg_match('/^[a-zA-Z]{2,64}$/i', $user_name)) {
        $this->errors[] = MESSAGE_NAME_INVALID;
    }
    elseif (strlen($user_surname) > 64 || strlen($user_surname) < 2) {
        $this->errors[] = MESSAGE_SURNAME_BAD_LENGTH;
    }
    elseif (!preg_match('/^[a-zA-Z]{2,64}$/i', $user_surname)) {
        $this->errors[] = MESSAGE_SURNAME_INVALID;
    }
    elseif (empty($user_email)) {
        $this->errors[] = MESSAGE_EMAIL_EMPTY;
    }
    elseif (strlen($user_email) > 64) {
        $this->errors[] = MESSAGE_EMAIL_TOO_LONG;
    }
    elseif (!filter_var($user_email, FILTER_VALIDATE_EMAIL)) {
        $this->errors[] = MESSAGE_EMAIL_INVALID;
    }
}
```

In questo caso controlliamo la connessione del database, e in seguito facciamo tutte le query che ci servono per la registrazione:

```
// finally if all the above checks are ok
} else if ($this->databaseConnection()) {
    // check if username or email already exists
    $query_check_user_name = $this->db_connection->prepare('SELECT user_email FROM users WHERE user_email = :user_email');
    $query_check_user_name->bindValue(':user_email', $user_email, PDO::PARAM_STR);
    $query_check_user_name->execute();
    $result = $query_check_user_name->fetchAll();

    // if username or/and email find in the database
    if (count($result) > 0) {
        $this->errors[] = MESSAGE_EMAIL_ALREADY_EXISTS;
    } else {
        // check if we have a constant HASH_COST_FACTOR defined (in config/hashing.php),
        // if so: put the value into $hash_cost_factor, if not, make $hash_cost_factor = null
        $hash_cost_factor = (defined('HASH_COST_FACTOR') ? HASH_COST_FACTOR : null);

        // crypt the user's password with the PHP 5.5's password_hash() function, results in a 60 character hash string
        // the PASSWORD_DEFAULT constant is defined by the PHP 5.5, or if you are using PHP 5.3/5.4, by the password hashing
        // compatibility library; the third parameter looks a little bit shitty, but that's how those PHP 5.5 functions
        // want the parameter: as an array with, currently only used with 'cost' => XX.
        $user_password_hash = password_hash($user_password, PASSWORD_DEFAULT, array('cost' => $hash_cost_factor));
        // generate random hash for email verification (40 char string)
        $user_activation_hash = sha1(uniqid(mt_rand(), true));

        // write new users data into database
        $query_new_user_insert = $this->db_connection->prepare('INSERT INTO users (user_name, user_surname, user_password_hash, user_email, user_registration_ip,
        user_registration_datetime) VALUES(:user_name, :user_surname, :user_password_hash, :user_email, :user_registration_ip, now())');
        $query_new_user_insert->bindValue(':user_name', $user_name, PDO::PARAM_STR);
        $query_new_user_insert->bindValue(':user_surname', $user_surname, PDO::PARAM_STR);
        $query_new_user_insert->bindValue(':user_password_hash', $user_password_hash, PDO::PARAM_STR);
        $query_new_user_insert->bindValue(':user_email', $user_email, PDO::PARAM_STR);
        $query_new_user_insert->bindValue(':user_registration_ip', $_SERVER['REMOTE_ADDR'], PDO::PARAM_STR);
        $query_new_user_insert->execute();

        // id of new user
        $user_id = $this->db_connection->lastInsertId();

        if ($query_new_user_insert) {
            $this->messages[] = MESSAGE_REGISTRATION_ACTIVATION_SUCCESSFUL;
        } else {
            $this->errors[] = MESSAGE_REGISTRATION_FAILED;
        }
    }
}
```

Con tutti questi pezzi di codici si arriva a questa pagina di registrazione, bisogna compilare tutti i campi per registrarsi correttamente.

## Registrati





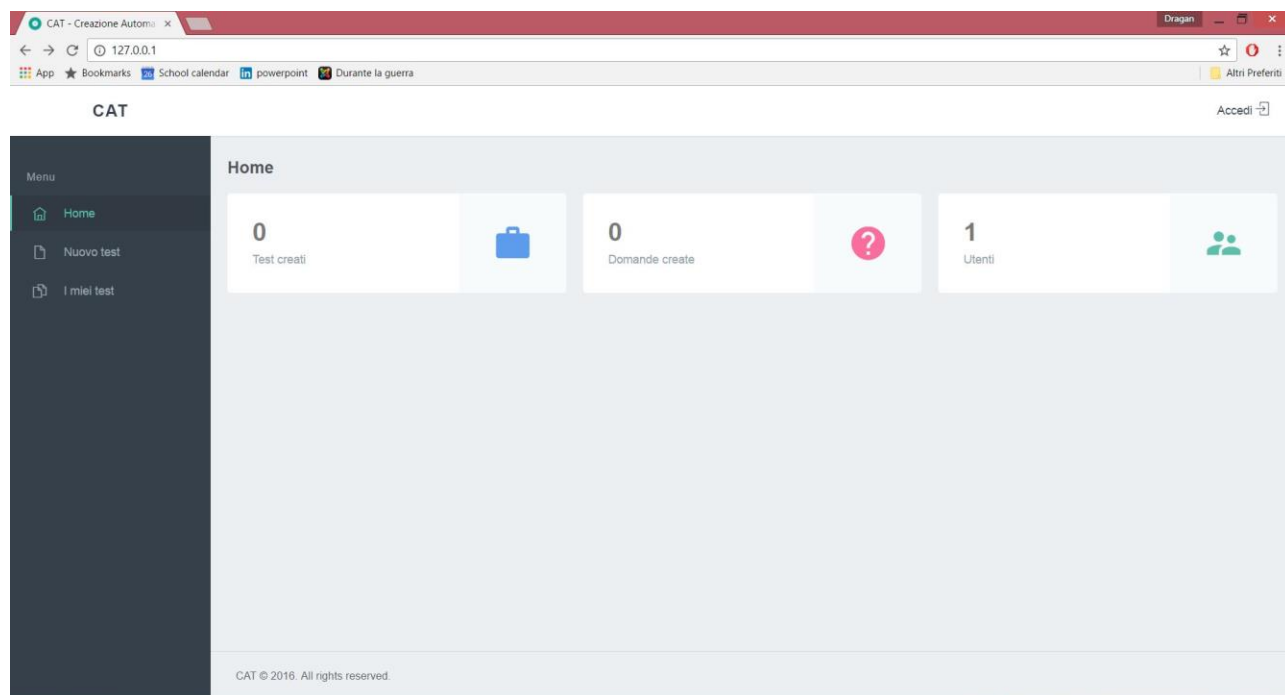

☐ Accetto i [Termini e le Condizioni](#)

**REGISTRATI**

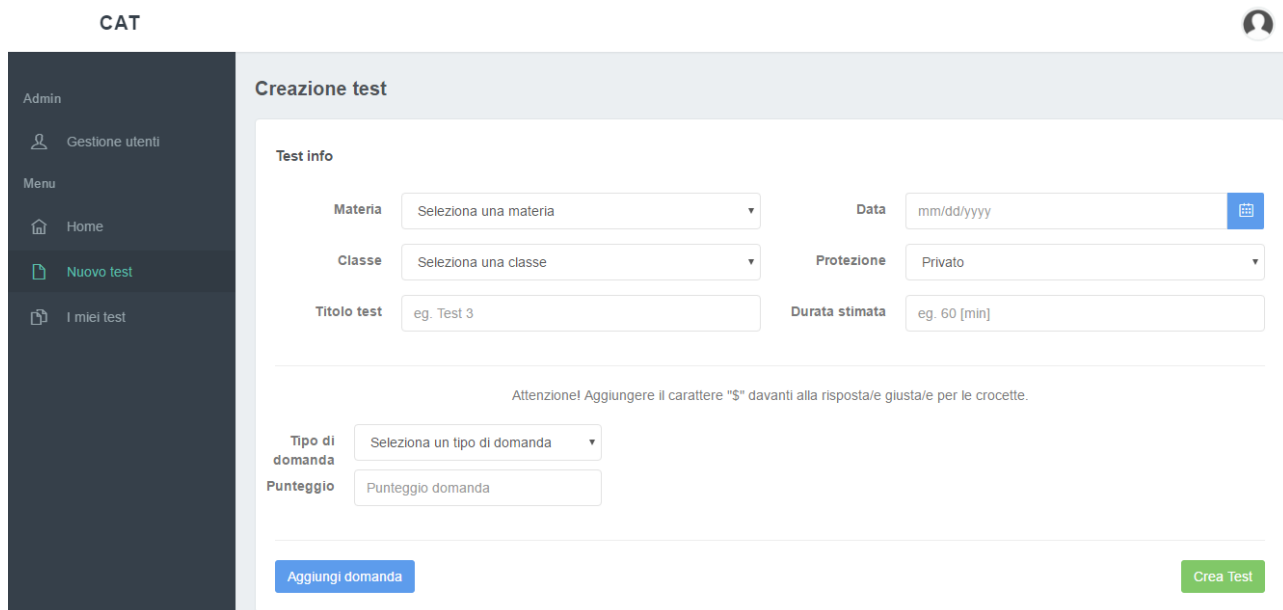
Hai già un account? [Accedi](#)

## 1.4 WEB

Come richiesto dal mandante, abbiamo realizzato una pagina per gli utenti che non sono ancora registrati. Questa schermata può essere visualizzata da chiunque, ovviamente per creare il test dovrà fare l'accesso, andando semplicemente a premere *Accedi* in alto a destra e compilare il modulo.



Una volta fatto l'accesso si possono creare le domande con il relativo test. Tutti i campi sono obbligatori, se si tralasciano via delle informazione non riuscirete a creare il test e il programma mostrerà dove sta il problema.



In seguito quando si saranno creati dei test, quest'ultimi si potranno vedere sotto "I miei test", in questa pagina saranno presenti tutti i test creati e si potranno scaricare in PDF:

CAT



Admin

Gestione utenti

Menu

Home

Nuovo test

**I miei test**

**I miei test**

Titolo test	Classe	Materia	Data	Protezione	Durata	PDF
Test Matematica 3	I3AA	Matematica	20.01.2017	Privato	60 minuti	Scarica
Test 2	I3AC	Mod 127	15.12.2016	Privato	90 minuti	Scarica

Come ultima pagina, solo per l'amministratore, sarà presente una gestione degli utenti. Qui l'admin potrà bannare i docenti, come si può vedere bastare premere "BAN" e l'utente verrà bannato.

CAT



Admin

**Gestione utenti**

Menu

Home

Nuovo test

I miei test

**Gestione utenti**

Utente	Email	Ban
Alessandro Narciso	alessandro.narciso@samtrevano.ch	
Dragan Jerkic	dragan.jerkic@samtrevano.ch	

## 1.5 PDF

Per la creazione del PDF abbiamo usato una libreria scaricata da internet, ovvero FPDF. Abbiamo usato i vari metodi messi a disposizione dalla libreria per creare tutta la formattazione e lo stile che poi avrà solamente bisogno dei dati presi dalla banca dati.

```

21 $pdf = new FPDF();
22 $pdf->AliasNbPages();
23 $pdf->AddPage();
24 $pdf->SetTitle(iconv('UTF-8', 'windows-1252', $title)." - ".iconv('UTF-8', 'windows-1252', $materia));
25
26 $pdf->Cell(0,5,"",0,1);
27
28 $pdf->SetFont('Arial','',15);
29 $pdf->Cell(0,8,"Nome Cognome:",'LTRB',2, "L");
30
31 $pdf->Cell(0,10,"",0,1);
32 $pdf->SetFont('Arial','',27);
33 $pdf->Cell(0,10,iconv('UTF-8', 'windows-1252', $title),0,1, "C");
34 $pdf->SetFont('Arial','',20);
35 $pdf->Cell(0,10,iconv('UTF-8', 'windows-1252', $materia),0,1, "C");
36 $pdf->Cell(0,8,"",0,1);
37
38
39 for($i = 0; $i < sizeof($domanda); $i++){
40
41     $pdf->SetFont('Arial','',15);
42     $pdf->MultiCell(0,8,iconv('UTF-8', 'windows-1252', $domanda[$i][0]),0,1);
43     $pdf->MultiCell(0,4,"",0,1);
44
45     for($o = 0; $o < sizeof($risposte[$i]); $o++){
46
47         if(!$domanda[$i][1]){
48             $pdf->SetFont('Arial','',15);
49
50             for($row = 0; $row < $risposte[$i][0]; $row++){
51                 $pdf->MultiCell(0,10,".....",0,1);
52             }
53
54         } else {
55             $pdf->SetFont('Arial','',15);
56             $pdf->MultiCell(0,8,"0 ".iconv('UTF-8', 'windows-1252', $risposte[$i][$o]),0,1);
57         }
58     }
59
60     $pdf->SetFont('Arial','',16);
61     $pdf->Cell(0,15,".../".iconv('UTF-8', 'windows-1252', $punteggio[$i]),0,1, "R");
62 }
63
64 $pdf -> Output();
65

```



Questo è un esempio di un semplice test generato con i due tipi di domanda e con la formattazione giusta.

Esempio Test 1 - Storia
1 / 1

Nome Cognome:

## Esempio Test 1

### Storia

Di che colore è il cavallo di napoleone?

☐ Bianco  
☐ Rosso  
☐ Nero

.../10

Spiegami con tue parole la seconda guerra mondiale

.....

.....

.....

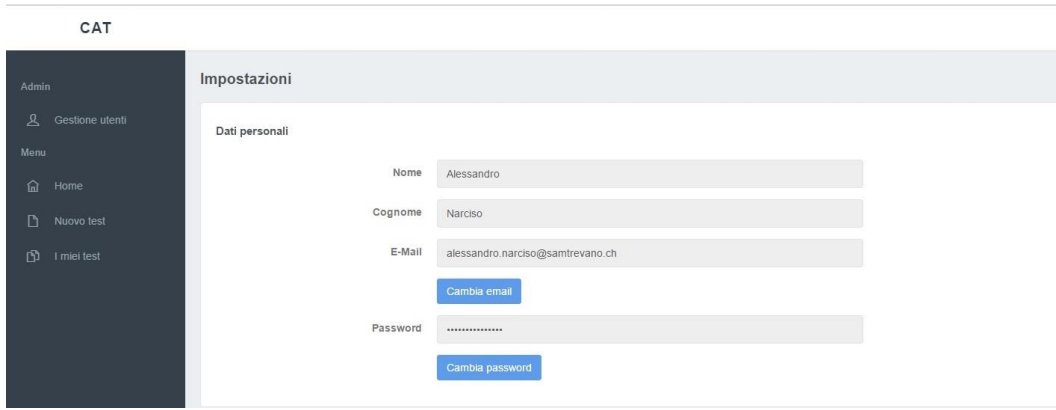
.....

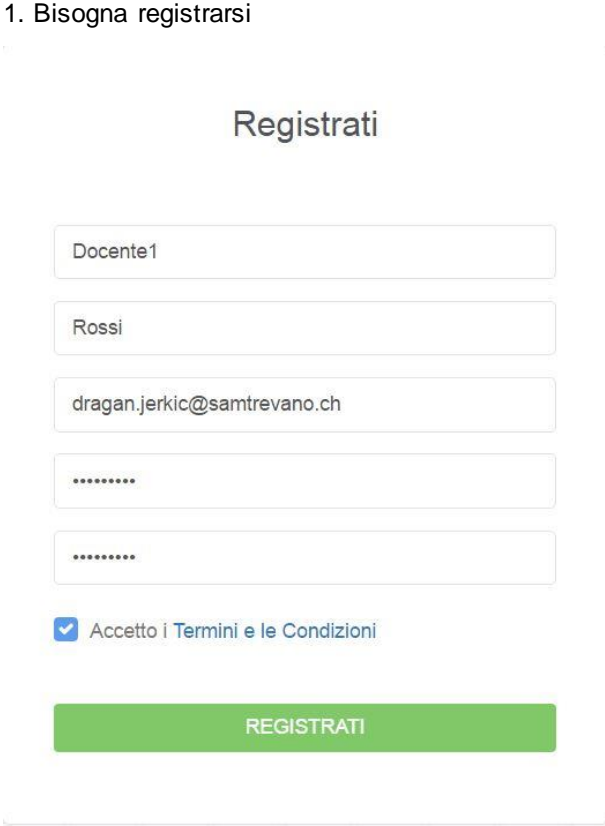
.....

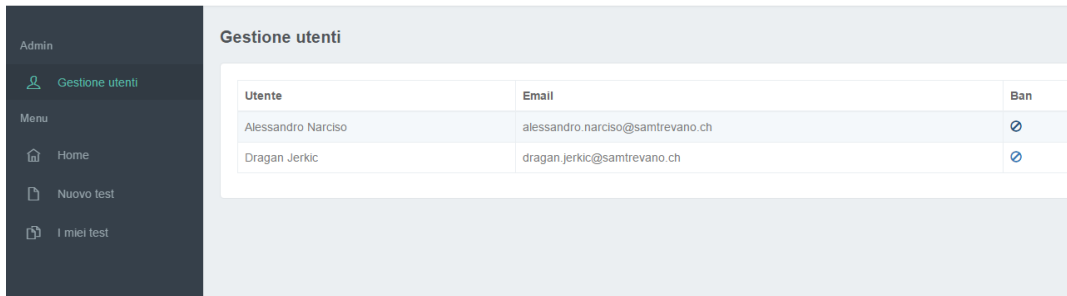
.../25

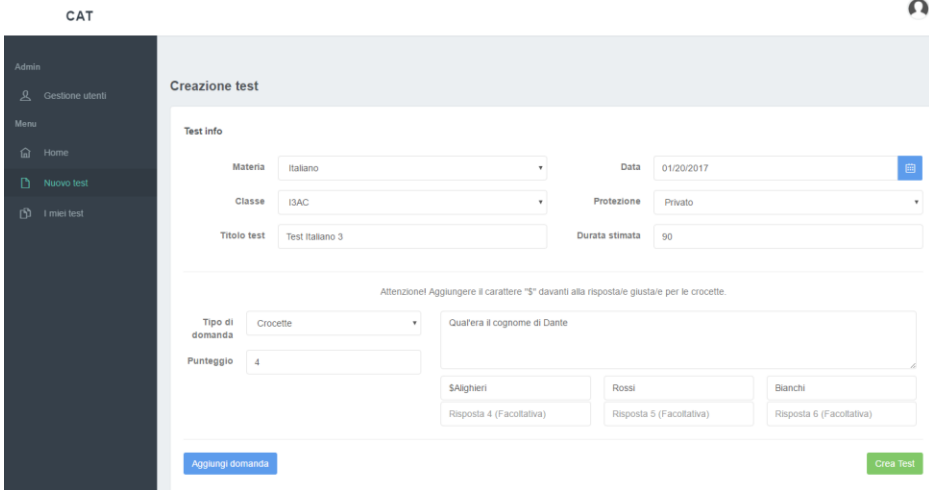
## 2 Test

### 2.1 Protocollo di test

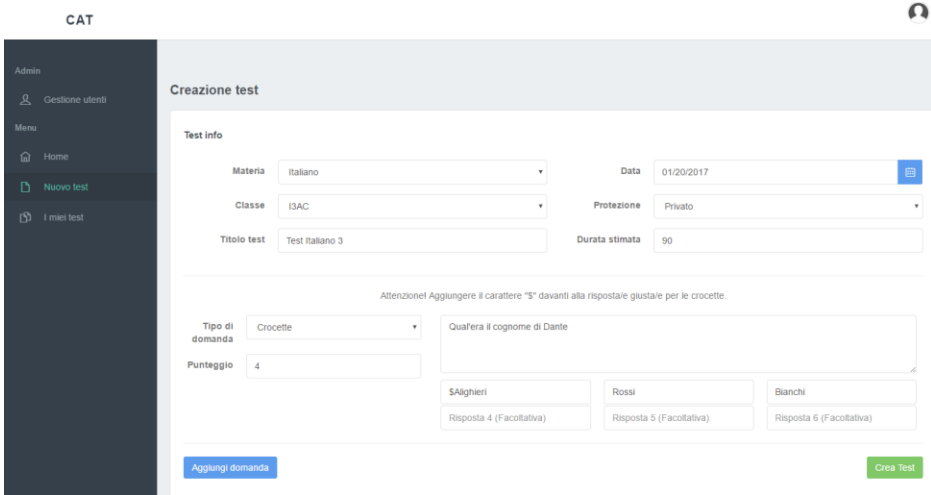
<b>Test Case:</b>	TC-001	<b>Nome:</b>	Test Login Admin
<b>Riferimento:</b>	REQ-001		
<b>Descrizione:</b>	Controlliamo se l'admin può accedere correttamente		
<b>Prerequisiti:</b>	-		
<b>Procedura:</b>	<p>1. Bisogna registrarsi</p> <p>2. Accedere alla pagina web</p> <p>3. Controllare I parametri</p> 		
<b>Risultati attesi:</b>	Accesso corretto		

<b>Test Case:</b>	TC-002	<b>Nome:</b>	Test Registrazione Docente
<b>Riferimento:</b>	REQ-001		
<b>Descrizione:</b>	Controlliamo se il docente può accedere correttamente		
<b>Prerequisiti:</b>	-		
<b>Procedura:</b>	<p>1. Bisogna registrarsi</p>  <p>2. Accedere alla pagina web</p>		
<b>Risultati attesi:</b>	Accesso corretto		

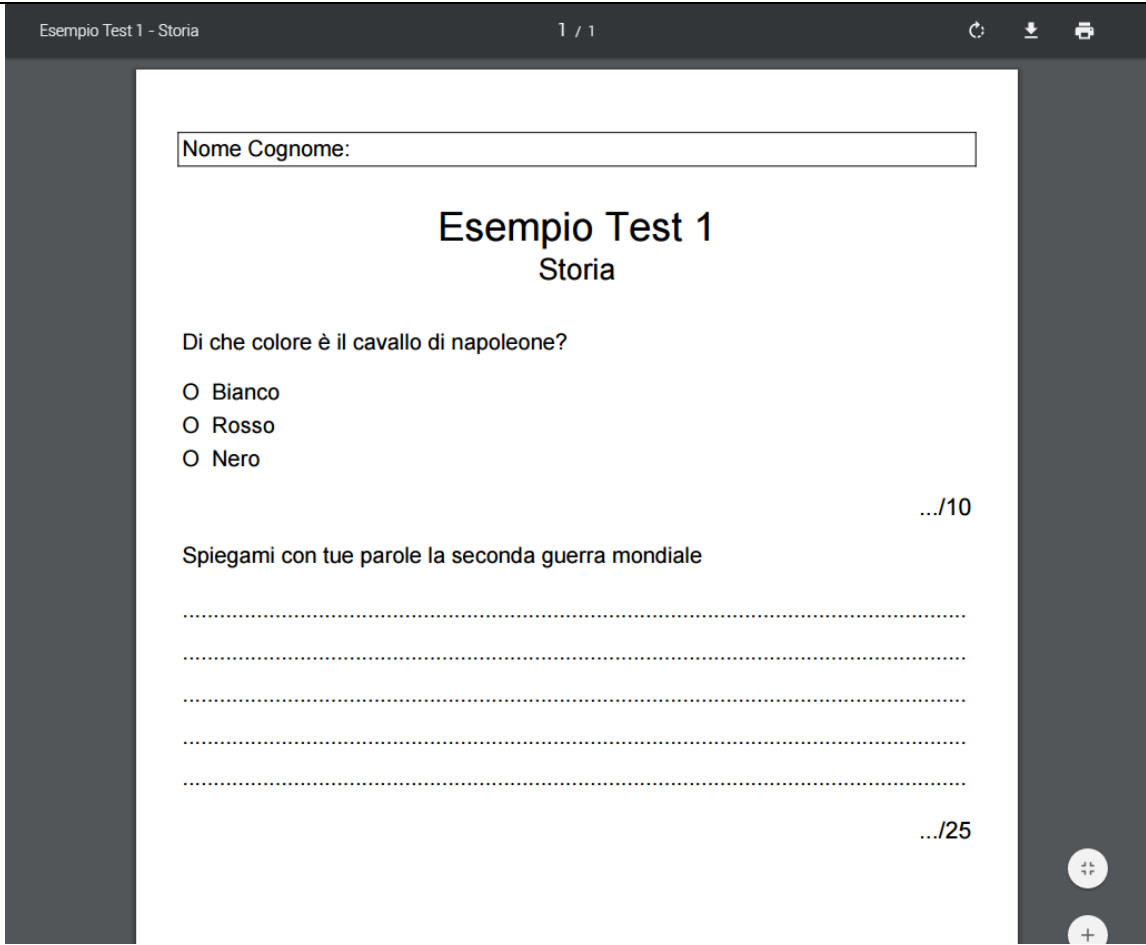
<b>Test Case:</b>	TC-003	<b>Nome:</b>	Test Gestione Utenti
<b>Riferimento:</b>	REQ-004		
<b>Descrizione:</b>	Controlliamo se l'amministratore può gestire i vari docenti		
<b>Prerequisiti:</b>	004		
<b>Procedura:</b>	<p>1. Accedere come amministratore</p> <p>2. Bisogna avere dei docenti connessi</p> <p>3. Scegliere quali tra quelli bannare</p> <p>CAT</p> 		
<b>Risultati attesi:</b>	Gestione corretta di tutti i docenti		

<b>Test Case:</b>	TC-004	<b>Nome:</b>	Creazione test Amministratore
<b>Riferimento:</b>	REQ-004		
<b>Descrizione:</b>	Controlliamo se l'amministratore può creare i test		
<b>Prerequisiti:</b>	002		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Inserire materia</li> <li>2. Inserire Data</li> <li>3. Inserire Classe</li> <li>4. Inserire Protezione</li> <li>5. Inserire Titolo</li> <li>6. Inserire Durata</li> <li>7. Scegliere il tipo di domanda (Crocette o Problema)</li> <li>8. Inserire Punteggio</li> <li>9. Inserire Risposte</li> <li>10. Crea Test</li> </ol>		
			
<b>Risultati attesi:</b>	Creazione avvenuta correttamente		



<b>Test Case:</b>	TC-005	<b>Nome:</b>	Creazione test Docente
<b>Riferimento:</b>	REQ-005		
<b>Descrizione:</b>	Controlliamo se il docente può creare i test		
<b>Prerequisiti:</b>	002		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Inserire materia</li> <li>2. Inserire Data</li> <li>3. Inserire Classe</li> <li>4. Inserire Protezione</li> <li>5. Inserire Titolo</li> <li>6. Inserire Durata</li> <li>7. Scegliere il tipo di domanda (Crocette o Problema)</li> <li>8. Inserire Punteggio</li> <li>9. Inserire Risposte</li> <li>10. Crea Test</li> </ol>		
			
<b>Risultati attesi:</b>	Creazione avvenuta correttamente		

**Implementazione e Test creazione automatica dei test**

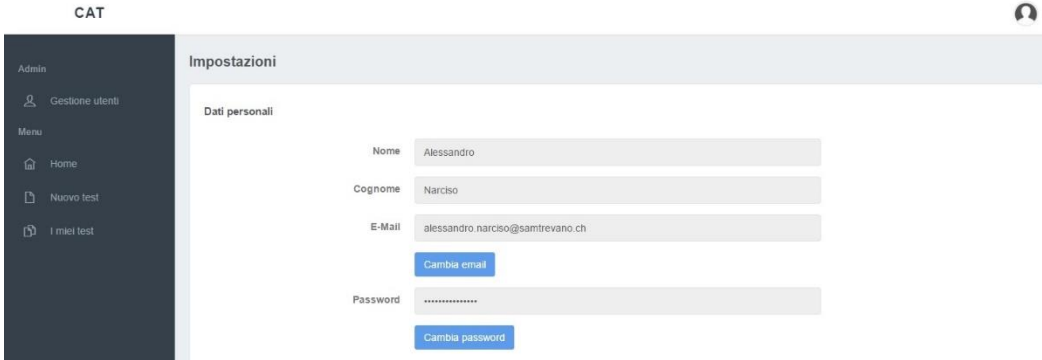
<b>Test Case:</b>	TC-006	<b>Nome:</b>	Creazione PDF
<b>Riferimento:</b>	REQ-003		
<b>Descrizione:</b>	Controlliamo se l'admin e i docenti possono creare un pdf		
<b>Prerequisiti:</b>	006		
<b>Procedura:</b>			
<b>Risultati attesi:</b>	Creazione avvenuta correttamente		

<b>Test Case:</b>	TC-007	<b>Nome:</b>	Cambio opzioni Admin
<b>Riferimento:</b>	REQ-004		
<b>Descrizione:</b>	Possibilità all'admin di cambiare la password e l'email		
<b>Prerequisiti:</b>	-		
<b>Procedura:</b>	1. Cambiare email 2. Cambiare Password		

**Implementazione e Test creazione automatica dei test**

	
<b>Risultati attesi:</b>	Cambiamenti messi in pratica

<b>Test Case:</b>	TC-008	<b>Nome:</b>	Cambio opzioni Docente
<b>Riferimento:</b>	REQ-005		
<b>Descrizione:</b>	Possibilità al docente di cambiare la password e l'email		
<b>Prerequisiti:</b>	-		
<b>Procedura:</b>	1. Cambiare email 2. Cambiare Password		
<b>Risultati attesi:</b>	Cambiamenti messi in pratica		



<b>Test Case:</b>	TC-009	<b>Nome:</b>	Scegliere opzioni per domanda
<b>Riferimento:</b>	REQ-002		
<b>Descrizione:</b>	Possibilità di scegliere la materia, le risposte, il punteggio e il tipo di domanda		
<b>Prerequisiti:</b>	001/2/3/4		
<b>Procedura:</b>	1. Inserire materia 2. Inserire Data 3. Inserire Classe 4. Inserire Protezione 5. Inserire Titolo 6. Inserire Durata 7. Scegliere il tipo di domanda (Crocette o Problema) 8. Inserire Punteggio 9. Inserire Risposte 10. Aggiungi Domanda o Crea Test		
<b>Risultati attesi:</b>	Possibilità di scegliere le opzioni correttamente		

## 2.2 Risultati test

Tabella riassuntiva dei test effettuati, in questa tabella si possono vedere i test passati e i test che hanno portato ad un errore nel progetto:

Riferimento Test	Risultato ottenuto
Test-001	Test superato
Test-002	Test superato
Test-003	Test superato
Test-004	Test superato
Test-005	Test superato
Test-006	Test superato
Test-007	Test superato
Test-008	Test superato
Test-009	Test superato

### 3 Consuntivi

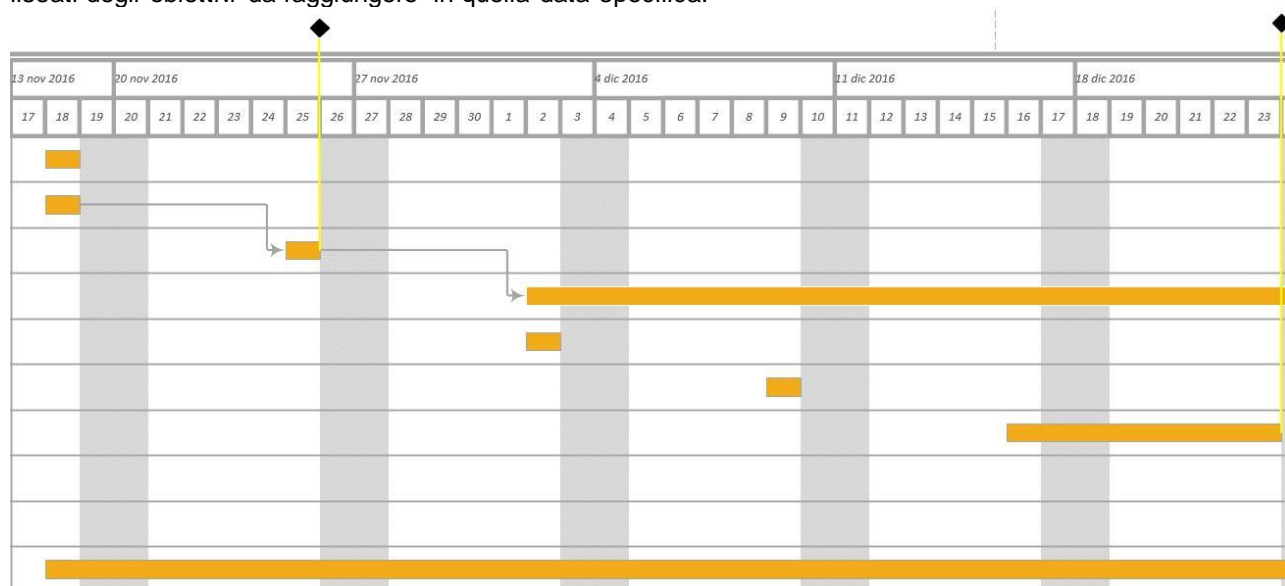
#### 3.1 Gantt consuntivo

Nel gantt consuntivo abbiamo inserito le varie attività effettuate durante tutta la fase del nostro progetto. Tra il consuntivo e il preventivo ci sono stati dei cambiamenti, abbiamo realizzato prima la parte di login e poi la creazione del database, mentre nel preventivo avevo previsto il contrario. Inoltre abbiamo aggiunto anche la creazione del PDF, che ci ha occupato molto più tempo del previsto siccome dovevamo capire come funzionava la libreria FPDF.

Nell'immagine seguente sono rappresentate le varie attività con le proprie date di inizio e di fine:

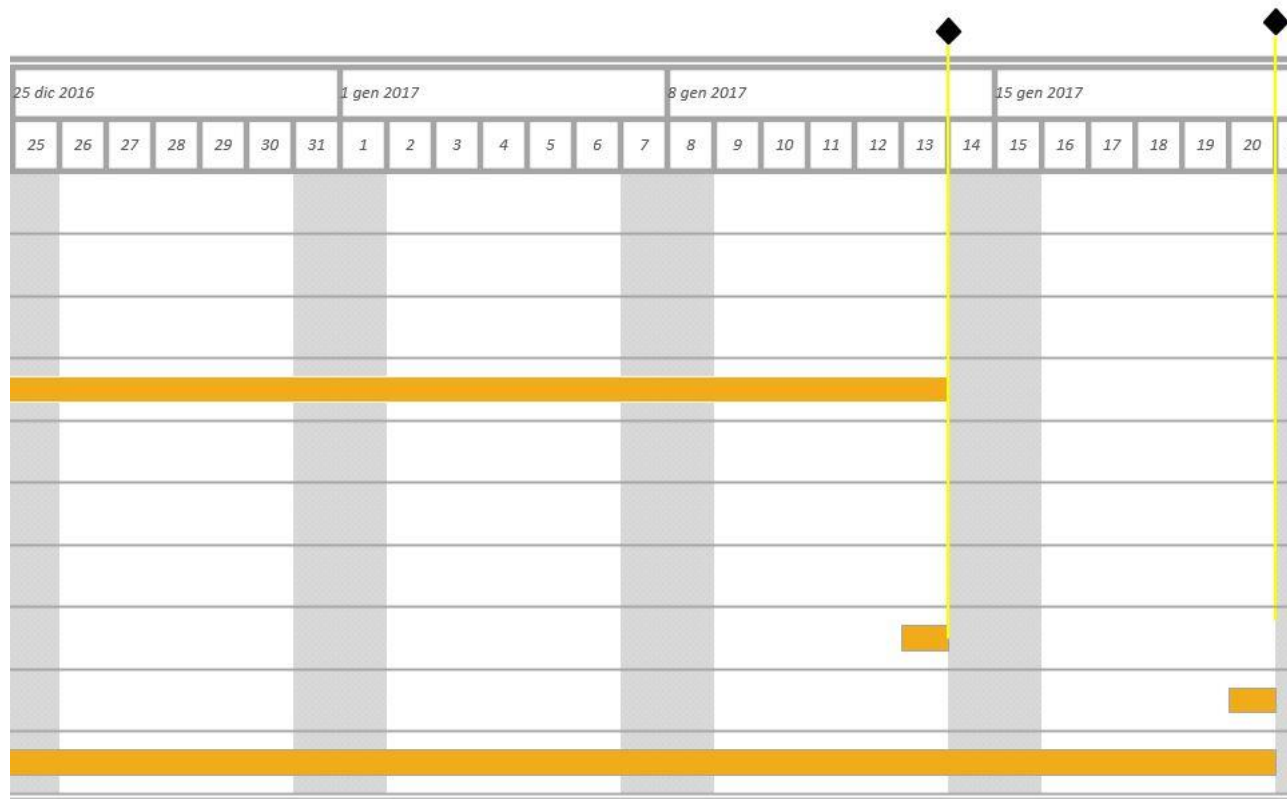
ID	Attività	Start	Finish
1	Analisi	18.11.2016	18.11.2016
2	Requisiti	18.11.2016	18.11.2016
3	Progettazione	25.11.2016	25.11.2016
4	Implementazione	02.12.2016	13.01.2017
5	Login	02.12.2016	02.12.2016
6	Creazione Database	09.12.2016	09.12.2016
7	Pagina WEB	16.12.2016	23.12.2016
8	Creazione PDF	13.01.2017	13.01.2017
9	Test	20.01.2017	20.01.2017
10	Documentazione	18.11.2016	20.01.2017

Nell'immagine è rappresentata la parte di pianificazione da noi realmente affrontata, in questa prima immagine sono raffigurate le durate delle nostre attività dal 18 novembre al 23 dicembre 2016. Qui si possono vedere effettivamente i cambiamenti rispetto al preventivo, inoltre abbiamo aggiunto un'altra milestone poiché ci siamo fissati degli obiettivi da raggiungere in quella data specifica:





In questa seconda immagine è raffigurata la pianificazione reale del nostro lavoro svolto dal 24 dicembre 2016 al 20 gennaio 2017. Si può vedere che abbiamo risparmiato del tempo per la pagina web, così abbiamo investito più tempo per la parte del PDF, e in quel punto abbiamo messo la terza milestone poiché è davvero importante realizzare il PDF.



### 3.2 Costo Consuntivo

Si può notare che il costo consuntivo e quello preventivo sono uguali, quindi abbiamo progettato bene i costi di questo progetto. Come per il preventivo l'unico costo che abbiamo avuto è quello delle risorse umane.

Risorse umane (45CHF/h)	Costo
Alessandro (42 h), Dragan (42 h)	3780 CHF

## **4 Conclusioni**

---

In conclusione possiamo dire che questo progetto ci è stato molto utile per consolidare e riprendere tutte le competenze imparate in questi anni. È stato bello lavorare in coppia poiché se avevamo un problema potevamo chiedere al compagno, inoltre è stato molto utile saper dividere i diversi compiti così tutti e due avevamo da lavorare, senza che uno stesse a guardare mentre l'altro lavorava. Questo progetto sarà molto utile per i docenti che lo utilizzeranno poiché non dovranno creare i loro test su word, ma potranno usare semplicemente il nostro progetto molto chiaro e molto facile da utilizzare.

### **4.1 Sviluppi futuri**

In futuro potremmo migliorare ulteriormente il nostro progetto, ad esempio aggiungendo un modulo di ricerca avanzata per cercare i vari test e le domande.

### **4.2 Considerazioni personali**

In questo progetto abbiamo imparato ad usare molti linguaggi assieme, ed è stato molto interessante mettere assieme le nostre competenze per realizzare questo progetto. Ci è piaciuto lavorare in coppia, abbiamo imparato a dividerci i vari compiti, così da poter lavorare contemporaneamente.

## **5 Sitografia**

---

- <http://www.w3schools.com/>
- <http://www.fpdf.org/>

## **6 Allegati**

---

Elenco degli allegati:

- Diari di lavoro
- Prodotto