Diario di lavoro

Titolo progetto	Gestione ore
Luogo	Trevano-Canobbio
Data	8 novembre 2019

Lavori svolti

Oggi ho proseguito con l'implementazione delle classi Controller, che fanno da ponte tra le classi Model e le view e viceversa.
Per essere più precisi, ho aggiunto un funzionamento alla funzione "details" del Controller 'Activities'. Per poterlo fare, però, ho dovuto aggiungere una funzione alla classe modello 'WorkHours', ovvero "`getWorkHoursByActivity`", che legge dalla tabella 'ore_lavoro' del database e restituisce un array di oggetti di tipo 'WorkHours' dove ogni elemento corrisponde a delle ore di lavoro svolte da una risorsa su un lavoro passato come parametro. I passaggi che giudico importanti sono mostrati di seguito:
Come prima cosa, la query che mi permette di leggere i dati dal database:

```
SELECT * FROM ore_lavoro WHERE nome_lavoro = :activity AND numero_ore
```

Dopodiché la creazione dell'oggetto da aggiungere all'array da ritornare (N.B. La variabile `$result` è un array associativo che contiene una delle righe ritornate dal'esecuzione della query, mentre `$baseResource` è un oggetto vuoto di tipo `Resource`):

```
//Get the resource assigned to the current work hours
$resource = $baseResource->getResourceByName($result["nome_risorsa"]);
//Create an object of type DateTime from the current work hours' date string
$date = DateTime::createFromFormat("Y-m-d", $result["data"]);
//Get the current work hours' hours number
$hoursNumber = intval($result["numero_ore"]);
//Create a new object of type WorkHours
$workHoursResult = new WorkHours($activity, $resource, $date, $hoursNumber);
```

La creazione di questa funzione mi ha permesso di completare la funzione "details" ottenendo tutti i dati da inserire nella pagina dei dettagli di un lavoro, che adesso ha questo aspetto (come esempio viene usato il lavoro 'Compleanno Susan'):

Trovo che sia importante far notare la logica che utilizzo per mostrare la pagina unicamente agli utenti che ne hanno il permesso, quindi gli amministratori e coloro che vi sono assegnati.

```
//Variable that defines if the user that is currently logged in can view the details pa
ge or not.
```

```php
//If the user is an administrator or they're assigned to the activity, they can view th
e page.
$canResourceView = ($loginResource->getRole() == "amministratore" ? true : false);
//To check that, loop all assigned resources and see if one of them is the same as the
one that logged in
for ($i = 0; !$canResourceView && $i < count($assignedResources); ++$i) {
    $assignedResource = $assignedResources[$i];
    if ($assignedResource->equals($loginResource)) {
        $canResourceView = true;
    }
}
//If the user cannot view the page, redirect them to the activities list
if (!$canResourceView) {
    $this->redirect("activities");
}
```

Ovviamento questo viene usato assieme a un controllo che la sessione sia stata avviata e il suo valore sia presente ed esista nel database.
Dopodiché mi sono occupato dell'azione "activities/new", che corrisponde alla funzione "new" della classe 'Activities'. Questa funzione prende i dati passati in POST dal form della pagina precedente e li usa per inserire una nuova riga all'interno della tabella 'lavoro' del database.

```php
//Check if the POST values are present
if (isset($_POST["nome"]) &&
    isset($_POST["note"]) &&
    isset($_POST["data_inizio"]) &&
    isset($_POST["data_consegna"]) &&
    isset($_POST["ore"])) {
    //Save all POST values to their respective variable
    $name = $this->sanitizeInput($_POST["nome"]);
    $startDate = $this->sanitizeInput($_POST["data_inizio"]);
    $startDate = DateTime::createFromFormat("Y-m-d", $startDate);
    $deliveryDate = $this->sanitizeInput($_POST["data_consegna"]);
    $deliveryDate = DateTime::createFromFormat("Y-m-d", $deliveryDate);
    $hoursNumber = intval($_POST["ore"]);
    $notes = $this->sanitizeInput($_POST["note"]);
    //Create a new object of type Activity with the POST values
    $activity = new Activity($name, $notes, $startDate, $deliveryDate, $hoursNumber);
    //Add the activity to the database
    $activity->addActivity($activity);
    //Redirect to the list of activities
    $this->redirect("activities");

}
```

Problemi riscontrati e soluzioni adottate
Ho visto che il numero di risorse assegnate ai vari lavori, mostrato sulla pagina contenente la lista delle attività, veniva ottenuto all'interno del foreach usato per popolare la pagina, quindi nella view, invece che nella classe Controller. Questo è ideologicamente sbagliato, quindi ho spostato quest'operazione nella classe Controller. Poi mi sono accorto che la validazione delle date su Internet Explorer permetteva di inserire anche date nel formato sbagliato, per esempio che non corrispondano al formato internazionale 'YYYY-mm-dd'. Ho dovuto modificare la logica di validazione del form in modo da far rispettare la seguente espressione regolare (ricavata dalla seguente pagina, https://forum.jquery.com/topic/jquery-validation-for-date): `/^\d{4}\-\d{1,2}\-\d{1,2}$/`

Punto della situazione rispetto alla pianificazione
Sono in anticipo rispetto alla pianificazione.

Programma di massima per la prossima giornata di lavoro
Portare avanti la documentazione della pianificazione e proseguire con l'implementazione delle classi Controller.