Diario di lavoro

Titolo progetto	Gestione ore
Luogo	Trevano-Canobbio
Data	12 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, mi sono occupato della classe 'ResourcesController', il cui nome è stato tradotto da quello pianificato, ossia 'RisorseController'. Ho completato tutte le sue funzioni, cominciando con "`index`", che controlla se l'utente ha eseguito l'accesso, in caso negativo lo rimanda alla pagina principale, mentre in caso positivo ne ottiene il ruolo. In caso esso sia amministratore ottiene tutte le risorse leggendole dal database, altrimenti legge solamente la risorsa corrispondente all'utente che ha eseguito l'accesso. Di seguito l'unica parte di codice che reputo importante, ovvero il controllo del ruolo e il conseguente ottenimento delle risorse da mostrare.

```
if ($_SESSION["userRole"] == Resource::ADMINISTRATOR_ROLE) {
    $resources = $baseResource->getAllResources();
} else {
    $resources = [$baseResource->getResourceByName($_SESSION["userName"])];
}
```

Poi sono passato alla funzione "aggiungi", tradotta in "add", che comincia controllando che l'utente abbia eseguito l'accesso, in caso contrario lo rimanda alla pagina principale, e che egli abbia il ruolo di amministratore, in caso contrario lo rispedisce alla pagina contenente la lista delle risorse. Dopodiché, la funzione controlla il metodo con cui è stata eseguita la richiesta, come faceva già prima delle modifiche di oggi; in caso il metodo sia POST, vengono letti i dati passati tramite il form e viene eseguito l'inserimento nel database. La porzione di codice seguente è quella che si occupa di leggere i dati del form della pagina precedente e li prepara per essere inseriti creandoci un oggetto di tipo Resource:

```
//Save all POST values to their respective variable
$name = $this->sanitizeInput($_POST["nome"]);
$cost = floatval($_POST["costo"]);
$password = $this->sanitizeInput($_POST["password"]);
$role = $this->sanitizeInput($_POST["ruolo"]);

//Create a new object of type Resource with the POST values
$resource = new Resource($name, $cost, $password, $role);
```

Infine ho concluso la classe 'Resource' portando a termine la sua funzione "dettagli", divenuta "details", la quale esegue il solito controllo per verificare che l'utente abbia eseguito l'accesso, in caso contrario lo rinvia alla pagina principale, poi legge dal database la risorsa di cui devono essere mostrati i dettagli basandosi sul suo nome, in caso il nome non esista rimanda alla lista di risorse, dopodiché controlla se l'utente sia amministratore o la risorsa richiesta, altrimenti lo rimanda alla lista di risorse per mancanza di permessi, infine ottiene i dati supplementari che devono essere mostrati sulla pagina, come i lavori a cui è assegnata. Ed è proprio quest'ultima cosa che viene eseguita grazie al seguente spezzone di codice:

```
$baseAssignment = new Assignment();
$assignedActivities = $baseAssignment->getActivitiesAssignedToResource($resource);
//Count the number of activities to which this resource is assigned
$assignedActivitiesCount = count($assignedActivities);
```

Sono poi passato all'implementazione della classe Controller 'Assegnazioni', tradotta in 'Assignments', che contiene una sola funzione: "assegna", divenuta "assign". Essa richiede come parametro il nome di un lavoro, usato per riempire il relativo campo all'interno della view, e contiene il codice che dapprima controlla, come al solito, se l'utente ha eseguito il login, altrimenti lo manda alla pagina principale, poi controlla che egli abbia il ruolo di amministratore, altrimenti lo rimanda alla pagina dei dettagli del lavoro il cui nome corrisponda alla stringa passata come parametro, poi controlla il metodo utilizzato per eseguire la richiesta: in caso esso sia GET ottiene tutte le risorse e le usa, assieme al nome del lavoro, per riempire la view 'assignments/assign', in caso dovesse essere POST vengono effettuati alcuni controlli, come quello per controllare se sono stati passati dati dal form e se i dati passati siano validi, infine viene creato un nuovo oggetto di tipo 'Assignment' e viene aggiunto al database.
Di seguito l'intera sezione del codice che gestisce le richieste di tipo POST:

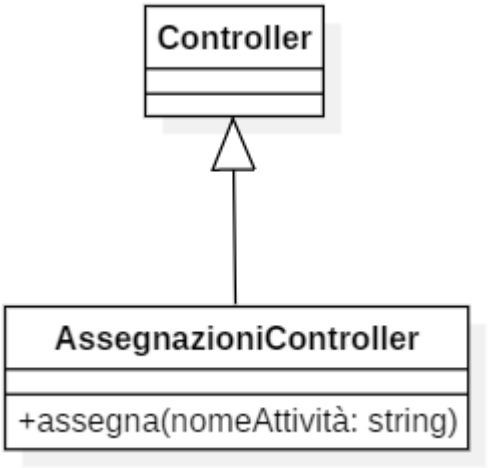
```
if (isset($_POST["lavoro"]) && isset($_POST["risorsa"])) {
    if ($_POST["lavoro"] === $activityName) {
        $resourceName = $this->sanitizeInput($_POST["risorsa"]);
        $activity = $baseActivity->getActivityByName($activityName);
        $resource = $baseResource->getResourceByName($resourceName);
        if (!is_null($activity) && !is_null($resource)) {
            $assignment = new Assignment($activity, $resource);
            if ($assignment->addAssignment($assignment)) {
                $this-
>redirect("activities", "details", [urlencode($activityName)]);
            }
        }
    }
}
```

Problemi riscontrati e soluzioni adottate

Fino ad ora, nell'implementazione delle varie parti del progetto stavo utilizzando i valori 'amministratore' e 'utente', relativi al ruolo che una risorsa può avere, in modo hard-coded, quindi utilizzando la stessa stringa dappertutto. Questo può portare a problemi in caso di cambiamento di questi due valori, perché sarebbe necessario modificarli per tutte le loro occorrenze. La soluzione è stata quella di aggiungere due costanti pubbliche alla classe Model 'Resource', '`ADMINISTRATOR_ROLE`' e '`USER_ROLE`', in modo da utilizzarle nel resto del codice in modo "dinamico".

```
public const ADMINISTRATOR_ROLE = "amministratore";
public const USER_ROLE = "utente";
/**
 * The valid values that can be set to field role.
 */
private const ROLE_VALUES = [self::ADMINISTRATOR_ROLE, self::USER_ROLE];
```

Mi sono poi accorto che alla progettazione della funzione “index” della classe Controller ‘AssegnazioniController’ manca un’indicazione del nome del lavoro al quale assegnare una risorsa. Ho quindi dovuto modificarla rimuovendo la funzione “index” e aggiungendo un parametro alla “assegna”:



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.