Diario di lavoro

Titolo progetto	Gestione ore
Luogo	Trevano-Canobbio
Data	25 ottobre 2019

Lavori svolti

Durante la giornata di oggi mi sono occupato di implementare le classi Model che si interfacciano al database ed eseguono le query su di esso.
Più di preciso ho proseguito con la creazione delle funzioni rimanenti per la classe 'WorkHours', scrivendo quella che nella pianificazione si chiamava "ottieniOreLavoro" e che ora è stata tradotta in "getWorkHours". Questa funzione restituisce tutte le ore di lavoro durante le quali una data risorsa ha lavorato a un certo lavoro. Ciò viene fatto grazie al codice sottostante:

```
/**
 * Returns all work hours, with date and number of hours, of a single resource on a sin
gle activity.
 * @param Activity $activity The activity on which the resource has worked.
 * @param Resource $resource The resource that has worked on the activity.
 * @return array An array of objects of type WorkHours, containing one element for each
 row of table 'ore_lavoro' of
 * the database whose activity and resource name correspond to the values of field 'nam
e' of the parameters passed
 * to this function.
 */
public function getWorkHours(Activity $activity, Resource $resource): array
{
    //The array that will contain the values to return
    $workHoursArray = [];
    //Check if activity and resource are valid
    if ($activity->isValid() &&
        $resource->isValid()) {
        //Get the activity's name
        $activityName = $activity->getName();
        //Get the resource's name
        $resourceName = $resource->getName();
        //Create the query
        $query = "SELECT * FROM ore_lavoro WHERE nome_lavoro = :activity AND nome_risor
sa = :resource";
        //Prepare the query
        $statement = $this->database->prepare($query);
        //Bind query placeholders to their respective values:
        //Bind placeholder :activity to value of 'name' field of parameter 'activity'
        $statement->bindParam(":activity", $activityName);
        //Bind placeholder :resource to value of 'name' field of parameter 'resource'
        $statement->bindParam(":resource", $resourceName);
        //Execute the statement
        if ($statement->execute()) {
```

```
            //Fetch the statement's results
            $models = $statement->fetchAll(PDO::FETCH_ASSOC);
            //If one or more rows have been returned
            if (count($models) > 0) {
                //Loop each returned row and for each add an object of type WorkHours t
o array workHours.
                foreach ($models as $model) {
                    //Get from the database the activity with the same name as activity
->getName().
                    $databaseActivity = $activity-
>getActivityByName($model["nome_lavoro"]);
                    //If the values of its fields are the same as the ones of parameter
 activity
                    if ($activity->equals($databaseActivity)) {
                        //Get from the database the resource with the same name as reso
urce->getName().
                        $databaseResource = $resource-
>getResourceByName($model["nome_risorsa"]);
                        //If the values of its fields are the same as the ones of param
eter resource
                        if ($resource->equals($databaseResource)) {
                            //All values correspond and the resource is actually assign
ed to the activity: create an object with values read from database.
                            $workHours = new WorkHours
                            (
                                $activity,
                                $resource,
                                DateTime::createFromFormat("Y-m-d", $model["data"]),
                                intval($model["numero_ore"])
                            );
                            //Add the object to the array
                            array_push($workHoursArray, $workHours);
                        }
                    }
                }
            }
        }
    }
    //Return array
    return $workHoursArray;
}
```

Dopodiché sono passato alla funzione “haLavoratoInData”, la quale è stata rinominata oltre a essere stata tradotta e il suo nuovo nome è “getWorkHoursNumberByDate”. Questa funzione si occupa di leggere il database e ritornare la somma di tutte le ore di lavoro di una risorsa su un lavoro in una determinata data.

Il suo codice è quello seguente:

```
/**
 * Gets the combined total number of hours during which a given resource has worked on
an activity on a given date.
 * @param Activity $activity The activity on which the resource has worked.
 * @param Resource $resource The resource that worked on the activity.
 * @param DateTime $date The date on which the work took place.
 * @return int The total number of hours during which the resource worked
 * on the activity on the given date, or 0 if no work hours were found.
 */
public function getWorkHoursNumberByDate(Activity $activity, Resource $resource, DateTi
me $date): int
{
    //Check if activity and resource are valid
    if ($activity->isValid() &&
        $resource->isValid()) {
        //Get the activity's name
        $activityName = $activity->getName();
        //Get the resource's name
        $resourceName = $resource->getName();
        //Format the DateTime object as a MySQL string
        $dateString = $date->format("Y-m-d");
        //Create the query
        $query =
            "SELECT SUM(numero_ore) FROM ore_lavoro " .
            "WHERE nome_lavoro = :activity AND nome_risorsa = :resource AND data = :dat
e";
        //Prepare the statement
        $statement = $this->database->prepare($query);
        //Bind query placeholders to their respective values:
        //Bind placeholder :activity to value of 'name' field of parameter 'activity'
        $statement->bindParam(":activity", $activityName);
        //Bind placeholder :resource to value of 'name' field of parameter 'resource'
        $statement->bindParam(":resource", $resourceName);
        //Bind placeholder :resource to MySQL-formatted value of parameter 'date'
        $statement->bindParam(":date", $dateString);
        //Execute the statement
        if ($statement->execute()) {
            //Fetch the statement's results
            $workHoursSum = $statement->fetch(PDO::FETCH_NUM);
            //If a row has been returned
            if ($workHoursSum) {
                return intval($workHoursSum[0]);
            }
        }
    }
    return 0;
}
```

Ho proseguito implementando una funzione “isValid”, che controlla se il valore dei campi dell’oggetto non sia nullo, se il numero di ore di lavoro sia maggiore di zero, se tutti i valori di lavoro e risorsa sono loro assegnati nelle rispettive tabelle e se il lavoro e la risorsa sono assegnati tra di essi nella tabella ‘assegna’:

```
/**
 * Checks if the values of the fields of this object of type Assignment are valid.
 * The values are valid when all of the following conditions are true:
 * - the value of fields 'activity', 'resource', 'date' and 'hoursNumber' is not null
 * - the value of field 'hoursNumber' is greater than zero
 * - the call to functions "isValid" of both activity and resource is true
 * - the values of the fields of both activity and resource are equal to those in the r
ow of their respective table
 * whose name is equal to the value of their 'name' field.
 * - the activity and resource are assigned inside table 'assegna' of the database
 * @return bool true if this instance of WorkHours is valid, false otherwise.
 */
public function isValid(): bool
{
    if (isset($this->activity) &&
        isset($this->resource) &&
        $this->activity->isValid() &&
        $this->resource->isValid()) {

        $baseAssignment = new Assignment();
        $databaseActivity = $this->activity->getActivityByName($this->activity-
>getName());
        $databaseResource = $this->resource->getResourceByName($this->resource-
>getName());

        return
            isset($this->date) &&
            isset($this->hoursNumber) &&
            intval($this->hoursNumber) > 0 &&
            $this->activity->equals($databaseActivity) &&
            $this->resource->equals($databaseResource) &&
            $baseAssignment->isAssigned($this->activity, $this->resource);

    }

    return false;

}
```

Sono andato avanti con la funzione “addWorkHours”, che aggiunge una riga alla tabella ‘ore_lavoro’ del database e il cui codice è simile a quello delle altre classi:

```
/**
 * Inserts a new row into table 'ore_lavoro' of database with data passed as parameter.
 * @param WorkHours $workHours An object of type WorkHours that contains the data to ad
d to the database.
 * @return bool true if the insert operation is successful, false otherwise.
 */
public function addWorkHours(WorkHours $workHours): bool
{
    //Check if the work hours to be added to the database are valid
    if ($workHours->isValid()) {
        //Insert a new row into table 'ore_lavoro' using inherited function "addModel".
        return $this->addModel([$workHours->activity->getName(), $workHours->resource-
>getName(), $workHours->date, $workHours->hoursNumber]);
    }
    return false;
}
```

Sono poi passato a “deleteWorkHours”, che fa il contrario e rimuove una riga dalla tabella ‘ore_lavoro’ del database. Il suo codice è quello visibile sotto:

```
/**
 * Deletes a record from the MySQL table 'ore_lavoro' where the name is equal to the na
me of an object of type WorkHours.
 * @param WorkHours $workHours The data of the assignment to delete.
 * @return bool true if the deletion is successful, false otherwise.
 */
public function deleteWorkHours(WorkHours $workHours): bool
{
    //Check if the work hours to be added to the database are valid
    if ($workHours->isValid()) {
        //Delete a row from table 'ore_lavoro' using inherited function "deleteModel".
        return $this->deleteModel([$workHours->activity->getName(), $workHours-
>resource->getName(), $workHours->date, $workHours->hoursNumber]);
    }
    return false;
}
```

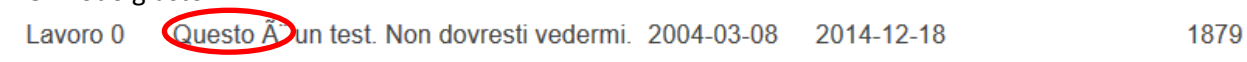
Ho terminato la classe con le funzioni getter per i campi ‘activity’, ‘resource’, ‘date’ e ‘hoursNumber’, di nome “getActivity”, “getResource”, “getDate” e “getHoursNumber”, rispettivamente. Queste funzioni sono indistinguibili l’una dall’altra, quindi di seguito è possibile trovare solo la funzione “getHoursNumber":

```
/**
 * Gets the number of hours the resource has worked on the activity for.
 * @return int The number of hours the resource has worked on the activity for.
 */
public function getHoursNumber(): ?int
{
    return $this->hoursNumber;
}
```

Questo conclude la fase di implementazione delle classi Model.

Problemi riscontrati e soluzioni adottate

Ho riscontrato un problema durante l'inserimento e l'eliminazione di un lavoro da e verso la tabella 'lavoro' del database: se ci sono caratteri speciali come la 'e' accentata (è o è) essi non vengono codificati nel modo giusto:



Ho risolto (grazie a questa domanda: **https://stackoverflow.com/questions/4475548/pdo-mysql-and-broken-utf-8-encoding/21373793**) aggiungendo quest'opzione come ultimo parametro passato al costruttore quando andavo a istanziare la connessione al database, in modo da forzargli come charset UTF-8:

```
array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8")
```

Questo ha risolto solo una parte del problema, infatti spesso non veniva inserito o rimosso un lavoro perché io comparavo due oggetti di tipo DateTime senza guardarne l'ora, per il cui valore veniva presa quella del sistema locale, quindi l'operatore `==` ritorna un valore positivo solo quando `startDate` e `deliveryDate` corrispondevano fino al millisecondo, cosa se non succedeva sempre. Per sistemare mi è bastato cambiare il confronto tra le due date, di cui mi serviva effettivamente solamente la data e non l'ora, nella funzione "isValid" in questo modo:

```
$activity->startDate->format("Y-m-d") == $this->startDate->format("Y-m-d") &&
$activity->deliveryDate->format("Y-m-d") == $this->deliveryDate->format("Y-m-d")
```

Ringrazio questa risposta (**https://stackoverflow.com/questions/7247259/get-the-year-month-day-from-a-datetime-in-php**) per l'aiuto.

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 realizzare la validazione del form di login di cui mi sono scordato nella fase di implementazione delle views.