Diario di lavoro

Titolo progetto	Gestione ore
Luogo	Trevano-Canobbio
Data	18 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 terminato l'implementazione della seconda classe Model, "Resource", aggiungendo le funzioni getter per i campi 'name', 'hourCost' e 'role': rispettivamente "getName", "getHourCost" e "getRole".
Di seguito, data l'uguaglianza nella logica delle tre funzioni, il codice di "getHourCost" come esempio:

```php
/**
 * Gets the cost per hour of this resource.
 * @return float The cost per hour of this resource.
 */
public function getHourCost(): float
{
    return $this->hourCost;
}
```

Poi ho iniziato a implementare la classe 'Assegnazione', che è stata tradotta in 'Assignment', partendo dalla funzione "ottieniTutteAssegnazioni", tradotta anch'essa in "getAllAssignments".
Come è possibile capire dal nome, questa funzione si preoccupa di leggere il database e ritornare un array di oggetti di tipo Assignment con i suoi dati.
Per farlo si avvale della superclasse 'Model', dalla quale eredita la funzione "getAllModels" che ritorna un array di array associativi con i dati letti dal database, per cui "getAllAssignments" si limita a prendere quei valori e convertirli in oggetti di tipo Assignment.

Di seguito il codice che compie quest'azione:

```
/**
 * Get all assignments reading their data from the database.
 * @return array An array containing a object of type Assignment for each line read fro
m the database.
 */
public function getAllAssignments(): array
{
    //Instantiates the array that will be returned.
    $assignments = [];
    //Get the database's data thanks to superclass 'Model'.
    $models = $this->getAllModels();
    //Instantiate an empty object of type Activity to use its methods
    $baseActivity = new Activity();
    //Instantiate an empty object of type Resource to use its methods
    $baseResource = new Resource();
    // Loop through each element read from the database and for each of them add an obj
ect of type Assignment with the data from the current element from models to array acti
vities.
    foreach ($models as $model) {
        //Get the activity with the same name as the current model.
        $modelActivity = $baseActivity->getActivityByName($model["nome_lavoro"]);
        //Get the resource with the same name as the current model.
        $modelResource = $baseResource->getResourceByName($model["nome_risorsa"]);
        //Add a new object of type Assignment with the model's data to the array.
        array_push($assignments, new Assignment($modelActivity, $modelResource));
    }
    return $assignments;
}
```

Dopodiché sono passato alla funzione "eAssegnato", che ho dovuto tradurre in "isAssigned". Questa funzione controlla che una risorsa sia assegnata a un lavoro guardando la tabella 'assegna' del database. Dopodiché ottiene il lavoro e la risorsa con il nome degli oggetti passati come parametro e controlla che i campi del lavoro letto dal database e quello passato come parametro abbiano gli stessi valori, infine esegue lo stesso controllo anche con le risorse.

Di sotto il codice della funzione "isAssigned":

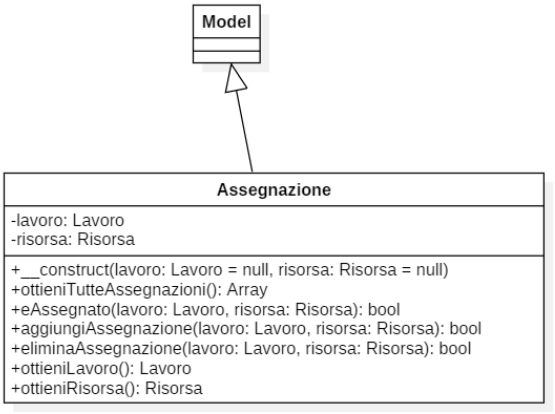
```
/**
 * Checks if an activity and a resource are associated in the database's 'assegna' tabl
e.
 * Note: this function should be used with parameters taken from a call to function "ge
tActivityByName" and
 * "getResourceByName" or "getAllActivities" and "getAllResources".
 * @param Activity $activity The activity from which to take the name to check if it is
 associated to a resource.
 * @param Resource $resource The resource from which to take the name to check if it is
 associated to an activity.
 * @return bool true if the two names are present on the same row in the table and if t
he values
 * of the two objects' fields exist in the same row of their respective tables, false o
therwise.
 */
```

```php
public function isAssigned(Activity $activity, Resource $resource): bool
{
    //Write the query that will be used to filter the database's data.
    $query = "SELECT * FROM assegna WHERE nome_lavoro = :activityName AND nome_risorsa
= :resourceName";
    //Prepare the query.
    $statement = $this->database->prepare($query);
    //Get the activity's name
    $activityName = $activity->getName();
    //Get the resource's name
    $resourceName = $resource->getName();
    //Bind the query's parameters to its placeholders.
    //Bind placeholder ':activityName" to value of field 'name' of parameter activity.
    $statement->bindParam(":activityName", $activityName);
    //Bind placeholder ':resourceName" to value of field 'name' of parameter resource.
    $statement->bindParam(":resourceName", $resourceName);
    //Execute the query
    $statement->execute();
    $isAssigned = false;
    //If a row has been returned
    if (count($statement->fetchAll()) === 1) {
        //Get from the database the activity with the same name as activity->getName().
        $databaseActivity = $activity->getActivityByName($activityName);
        //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 resource-
>getName().
            $databaseResource = $resource->getResourceByName($resourceName);
            //If the values of its fields are the same as the ones of parameter resourc
e
            if ($resource->equals($databaseResource)) {
                //All values correspond and the resource is actually assigned to the ac
tivity
                $isAssigned = true;
            }
        }
    }
    //If the parameters are equal to their respective database rows, that means the res
ource is assigned to the activity.
    return $isAssigned;
}
```

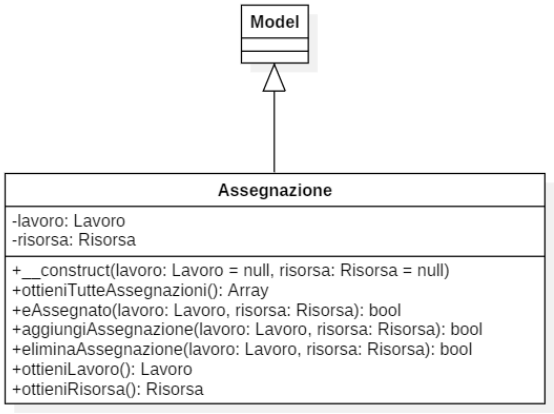
Problemi riscontrati e soluzioni adottate

Mi sono accorto che nella progettazione della classe Model 'Assegnazione' dopo una delle modifiche recenti il rettangolo della classe 'Assegnazione' non è allineato verticalmente con quello della superclasse 'Model'. Quindi ho dovuto spostarlo in modo che i due rettangoli siano centrati tra loro.

Prima l'immagine era questa:



Oggi l'ho modificata in questa:



Verso la fine della giornata mi sono reso conto che ho dovuto controllare l'uguaglianza tra due istanze della stessa classe quindi, per motivi di sicurezza, ho dovuto aggiungere una funzione "equals" all'interno delle sottoclassi di Model che sono state implementate fino a questo punto. Questa funzione controlla che il valore dei campi sia uguale tra i due oggetti. Ad esempio, di seguito quella della classe Resource:

```
/**
 * Checks if an object of type Resource's fields have the same value as the fields of t
his instance of Resource.
 * @param Resource $resource The object against which to compare.
 * @return bool true if the two object's fields have the same value, false otherwise.
 */
public function equals(Resource $resource)
{
    return
        $resource->name === $this->name &&
        $resource->hourCost === $this->hourCost &&
        $resource->password === $this->password &&
        $resource->role === $this->role;
}
```

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 la realizzazione delle classi Model che si interfacciano al database.