**Diario di lavoro -31.01.2017**

|  |  |
| --- | --- |
| Luogo | Trevano |
| Data | 31.01.2017 |

|  |
| --- |
| Lavori svolti |
| Per prima cosa ho continuato a lavorare sulla registrazione dell’utente aggiungendo i campi rimasti(surname, phone e birthday). Per farlo ho utilizzato lo stesso procedimento visto nel diario di lavoro del 30.01.2017 fatto per il campo *type*.  Dato che quando un utente si registra deve aspettare la conferma di un admin ho deciso di assegnare al campo *stato* un valore di default.  Per fare ciò bisogna andare sotto *database/migration* e selezionare il file relativo alla tabella users e aggiungere il metodo *default()* al campo nel seguente modo:  $table->integer('state')->default(0);  In seguito ho utulizzato la guida al seguente link: <https://scotch.io/tutorials/a-guide-to-using-eloquent-orm-in-laravel> per creare la struttura del database con le relazioni.  La prima cosa da fare è creare la tabella che mette in relazione un utente con un progetto in modo che i docenti o gli admin possano gestirli, per fare ciò creare un nuovo file di migrazione per la tabella tramite il prompt dei comandi:  php artisan make:migration create\_manages\_table –create=manages  Questo comando crea un nuovo file di migrazione contenente dei campi di default, cambiarli inserendo i nomi delle foreign key:  public function up() {  Schema::*create*('manages', function (Blueprint $table) {  $table->increments('id');   $table->integer('user\_id');  $table->integer('project\_id');   $table->timestamps();  }); }  Successivamente bisogna creare delle funzioni per specificare che *manages* è un tabella che mette in relazione *users* e *projects*. Per fare ciò nel file *app/User.php* aggiungere la seguente funzione:  public function projects(){  return $this->belongsToMany('App\Project','manages','user\_id','project\_id'); }  Il metodo *belongsToMany* serve per creare la relazione N:N e prende 4 parametri:   1. L’oggetto con cui mettersi in relazione 2. La tabella relazionale 3. Il proprio id nella relazione 4. L’id dell’oggetto in relazione   Fare una funzione simile nel file *app/Project.php*:  public function users(){  return $this->belongsToMany(‘App\User','manages','project\_id','user\_id'); }  Inoltre ricordarsi di estendere *Model* nella classe *Project.php*:  class Project extends Model  Non è necessario farlo per *User.php* poichè è già esteso per via del login/registrazione:  class User extends Authenticatable  Successivamente bisogna modificare il file *routes/web.php* e aggiungere:  Route::*get*('eloquent', function() {  return view('eloquent')->with('users', User::*all*()); });  In seguito bisogna creare la pagina per visualizzare la lista delle assegnazioni che bisogna chiamare *nome.blade.php*. È una pagina con del semplice html e bootstrap che stampa a schermo l’:  <table class="table table-striped">  <thead>  <td>Allievo</td>  <td>Progetto</td>  <td>Numero</td>  </thead>  @foreach ($users as $user)  <tr>  <td>{{ $user->name }} {{ $user->surname }}</td>  @foreach ($user->projects as $project)  <td>{{ $project->name }}</td>  <td>{{ $project->number }}</td>  @endforeach  </tr>  @endforeach </table>  La @ richiama php, questo rende il codice pulito, facilmente leggibile e decisamente più veloce da scrivere. Inoltre {{ richiama un echo che stampa il valore al suo interno.  Il risultato del codice precedente è il seguente: |

|  |
| --- |
| Problemi riscontrati e soluzioni adottate |
| Ho avuto problemi poichè non estendevo *Model* nella classe *Project.php* e perchè nelle funzioni per associare le tabelle scrivevo *Project* o *User* senza metterci davanti *App\*. |

|  |
| --- |
| Punto della situazione rispetto alla pianificazione |
| In linea con la pianificazione |

|  |
| --- |
| Programma di massima per la prossima giornata di lavoro |
| Studio di laravel/implementazione |