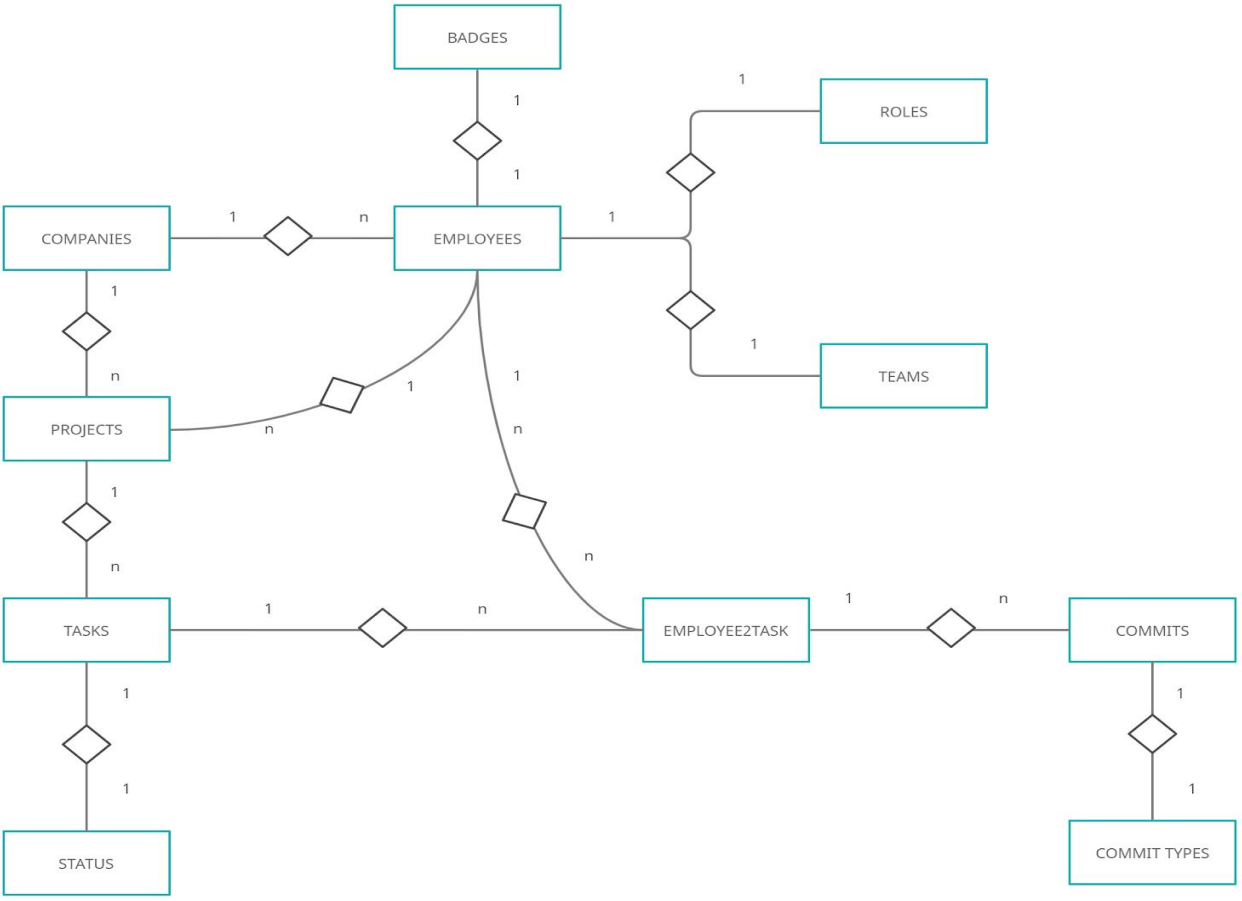
WEBFORMAT PROGETTO TEST

Guideline e commenti del progetto di test “Ticket System”

Modellazione ER secondo le caratteristiche indicate



La modellazione del ticket system sopra riportato prevede che:

1. Un'azienda informatica ha degli impiegati

2. Ogni impiegato ha un ruolo (CEO, PM, DEV)

3. Ogni impiegato tranne il CEO è associato ad un team

La chiave esterna (team_id) presente nella tabella employees, per un impiegato con ruolo CEO, dovrà essere impostata a NULL negli inserimenti via software.

4. * Ogni impiegato ha un badge che usa per entrare / uscire dall'ufficio e registra i tempi di lavoro

Oltre a prevedere l’entità “Badges” sopra riportata, per tenere traccia ed avere uno storico delle entrate e delle uscite dall’ufficio di ogni impiegati, bisognerebbe prevedere una nuova entità “Stamping” in relazione ad Employees.
Ogni Employee può timbrare N volte per entrate e uscite e, ciascuna di essa è strettamente correlata ad un employee.
Inoltre, è possibile prevedere una relazione 1-1 fra “Stamping” e “Badges” ottenendo per ogni timbratura il badge correlato.

5. L'azienda lavora su progetti che il CEO assegna ad un PM
All’inserimento di record nella tabella progetti via software si controllerà che la chiave esterna (employee_id), presente nella tabella progetti, si riferisca ad un employee con ruolo PM.

6. Il PM per il progetto crea dei task che hanno una descrizione, uno status e una deadline (data entro la quale il task deve essere chiuso)

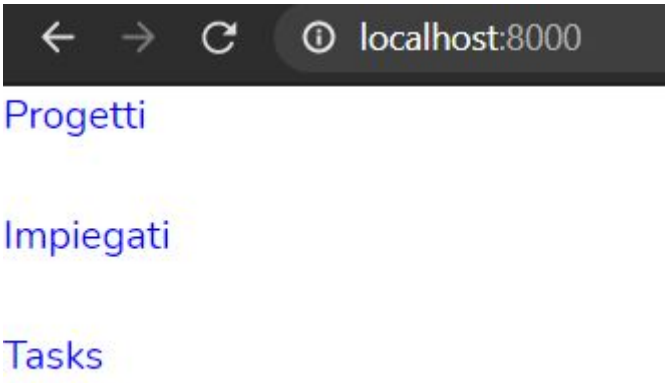
7. Un task può essere assegnato ad uno o più sviluppatori (impiegato con ruolo DEV)
Nella tabella di intersezione employee2task via software si controllerà che la chiave esterna (employee_id) si riferisca ad un employee con ruolo DEV.

8. * Un task può avere dei commit (messaggi o note) che sono fatti da uno sviluppatore
La tabella commit è in relazione con la tabella di intersezione “employee2task” in modo tale da ottenere, all’interno dell’entità Commit, la chiave “employee2task_id” che permette, a livello software, di accedere più facilmente non solo ai task ma anche agli employees.

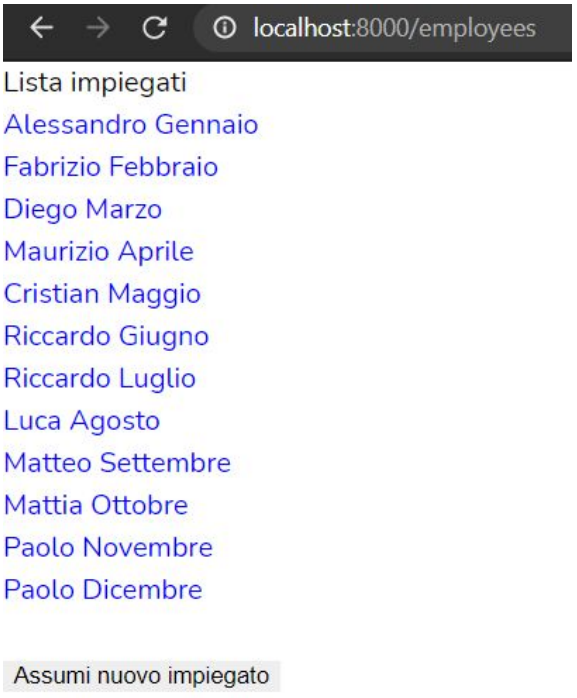
9. Il CEO può assumere impiegati PM o DEV
Via software si controllerà che ogni aggiunta fatta alla tabella employee sia eseguita da utente con ruolo CEO.

Sviluppo software

Per lo sviluppo del progetto ho creato un'interfaccia web, utilizzando PHP con framework Laravel.
Accedendo alla default route, sono presenti 3 link che indirizzano rispettivamente alle pagine indicate.

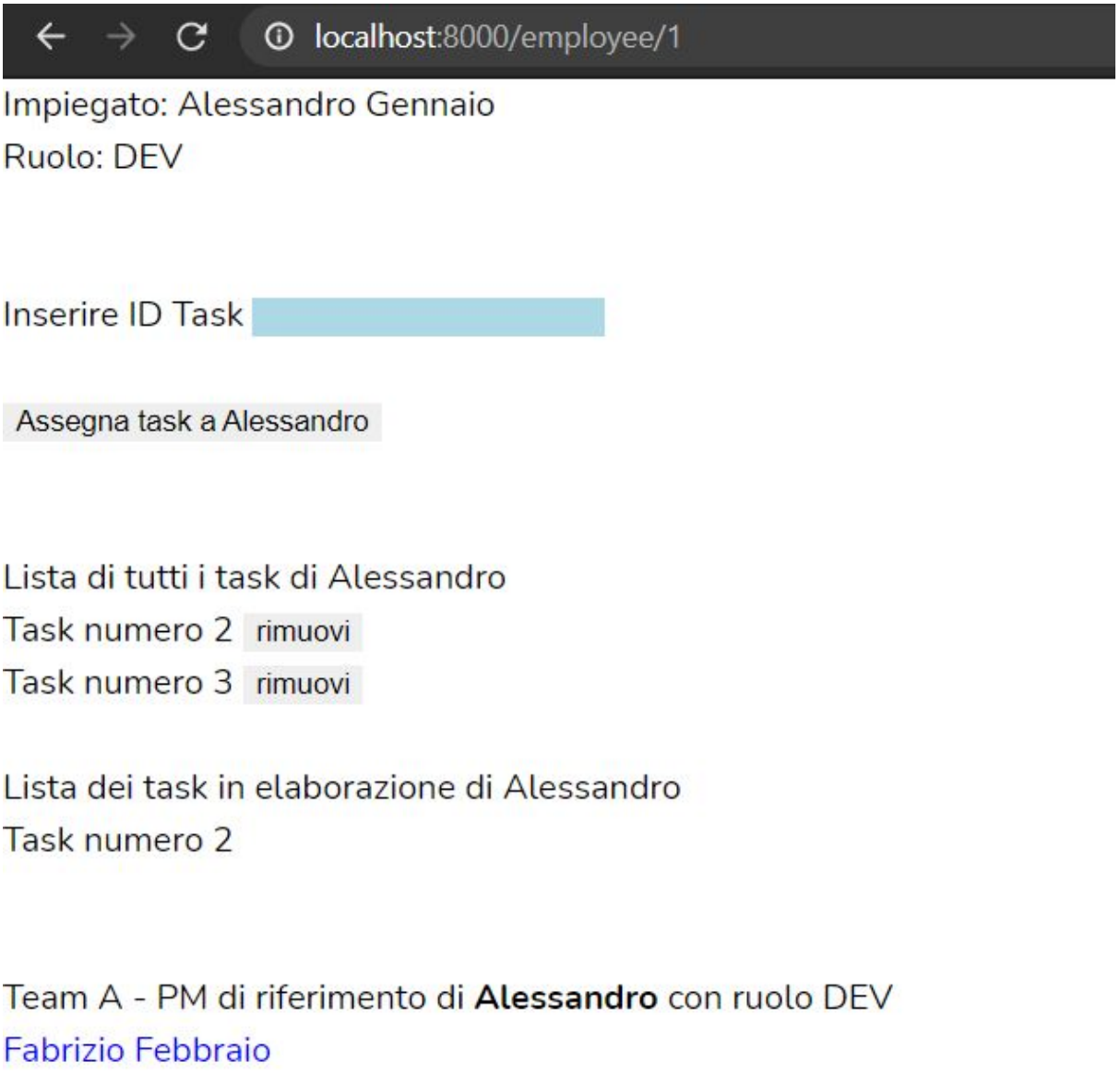


Accedendo alla sezione impiegati, è presente una lista di tutti gli impiegati dell'azienda.

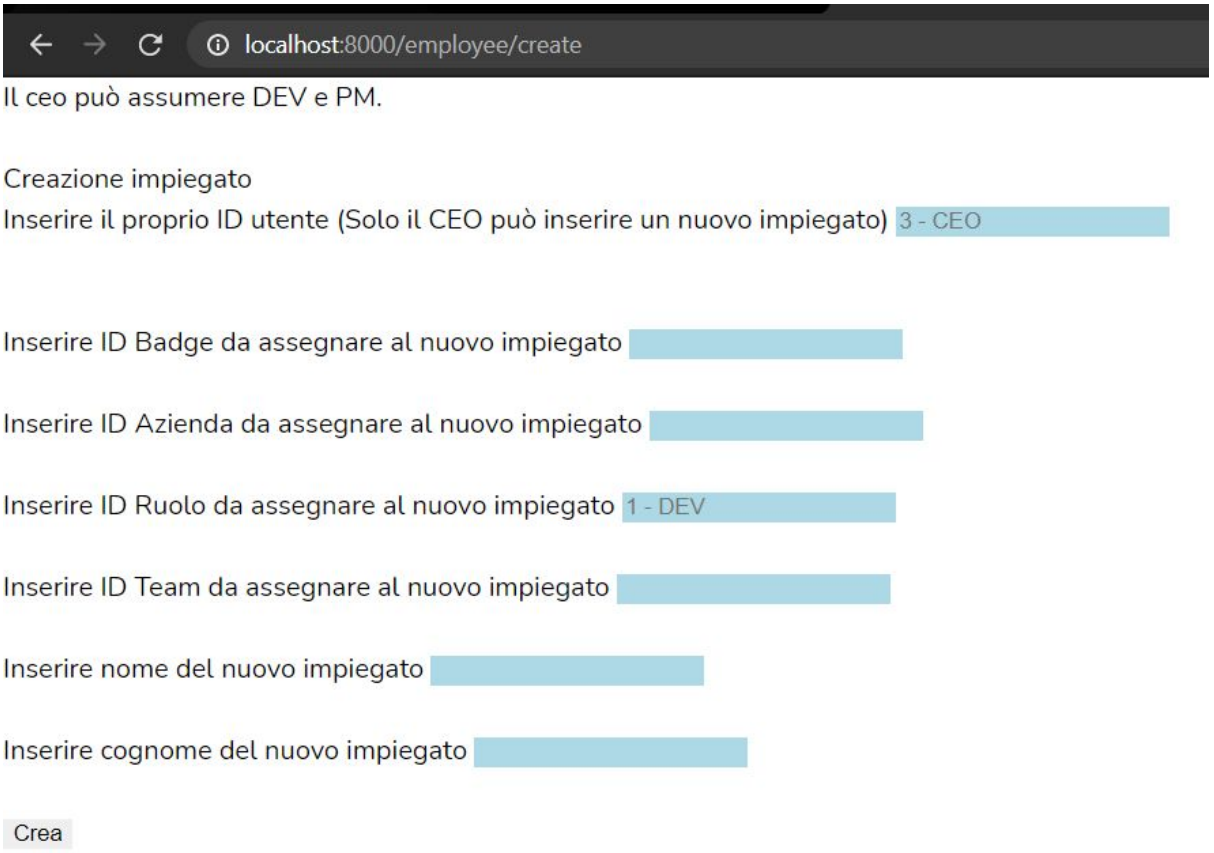


Al click su ciascun impiegato, si entra nella schermata di dettaglio di esso e, nel caso si tratti di uno sviluppatore (impiegato con ruolo DEV), sono previste le funzionalità di assegnazione (1) e rimozione di tasks (2). Inoltre, sotto alla lista dei task assegnati ad uno sviluppatore, è presente la sezione che mostra solamente i task che sono in stato di "In elaborazione" (3).

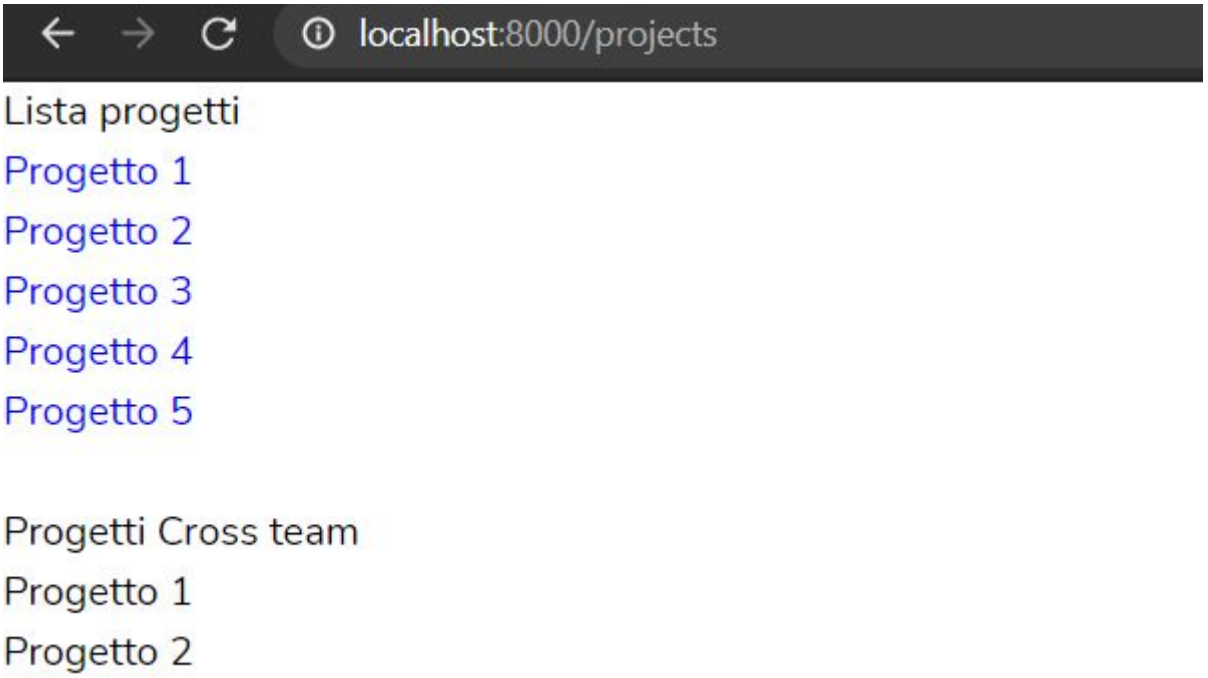
Infine, viene mostrato il/i PM di riferimento dello sviluppatore (6) in questione (il/i PM che fanno parte dello stesso team dell'impiegato)



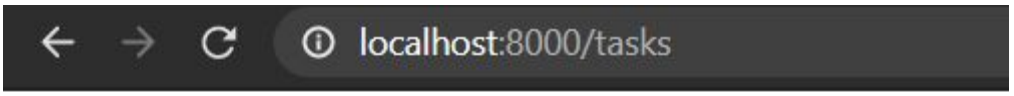
Dalla lista degli impiegati (/employees) è possibile assumere un nuovo impiegato ed assegnarlo ad un team(5).



Nella lista dei progetti (/projects) viene mostrata la lista di tutti i progetti dell'azienda e, più in basso, i progetti cross-team (6) ovvero progetti che hanno tasks assegnati a sviluppatori di almeno 2 team diversi



Analogamente alla sezioni progetti, nella lista dei task (/task) vengono mostrati tutti i task e, in particolare, i task che hanno sforato la deadline con i relativi sviluppatori coinvolti



Lista tasks

Task numero 1

Task numero 2

Task numero 3

Task numero 4

Task numero 5

Lista tasks che hanno sforato la deadline

Task numero 1

["Paolo Dicembre"]

Task numero 2

["Alessandro Gennaio","Maurizio Aprile"]

Task numero 3

["Mattia Ottobre","Alessandro Gennaio"]

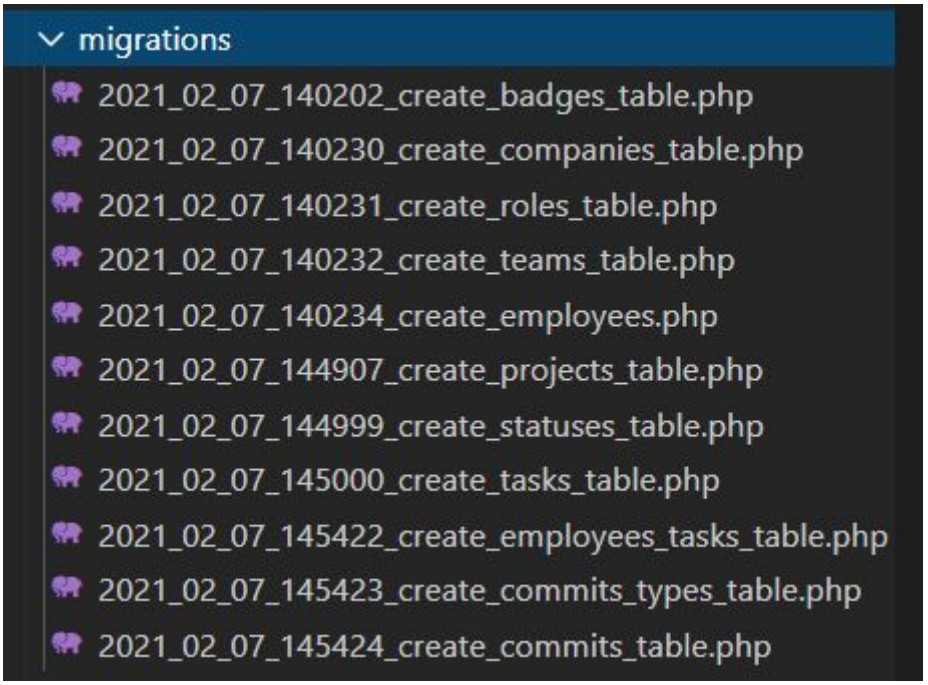
Per testare l'interfaccia web è necessario lanciare il comando *php artisan serve* a terminale per effettuare lo start del server del development server.

Database

Nel file .env vengono specificati i parametri di connessione al database.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=webformat
DB_USERNAME=root
DB_PASSWORD=
```

Una volta creato il database, lanciando da terminale il comando *php artisan migrate* verranno eseguite ordinatamente le migrations sotto riportate permettendo così la creazione automatica di tutte le tabelle ed i vincoli fra esse.



Ciascuna di esse contiene due funzioni: la prima (run) che creerà la tabella indicata con la chiave primaria id, i campi scelti e le chiavi esterne ad essa correlate.

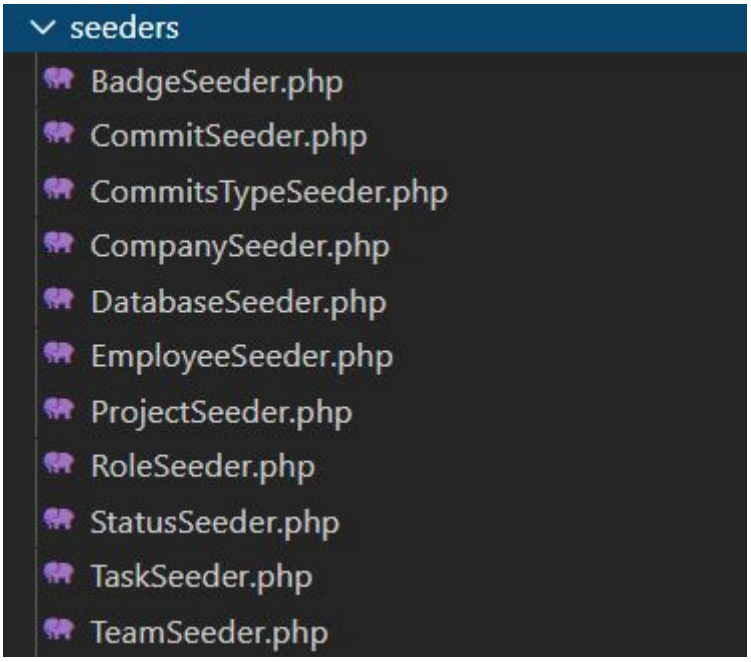
Invece, la seconda funzione (down), permette di effettuare il drop della tabella in questione. Laravel crea automaticamente una tabella a database chiamata migrations in modo tale da permettere il rollback ad una migrazione precedente tramite il comando *php artisan migrate:rollback*

L'esempio mostrato sotto si riferisce alla migration della tabella employees.

```php
Schema::create('employees', function (Blueprint $table) {
    $table->id();
    $table->unsignedBigInteger('badge_id');
    $table->unsignedBigInteger('company_id');
    $table->unsignedBigInteger('role_id');
    $table->unsignedBigInteger('team_id');
    $table->string("name");
    $table->string("surname");


    $table->foreign('badge_id')
        ->references('id')->on('badges')
        ->onUpdate('cascade')
        ->onDelete('cascade');

    $table->foreign('company_id')
        ->references('id')->on('companies')
        ->onUpdate('cascade')
        ->onDelete('cascade');

    $table->foreign('role_id')
        ->references('id')->on('roles')
        ->onUpdate('cascade')
        ->onDelete('cascade');

    $table->foreign('team_id')
        ->references('id')->on('teams')
        ->onUpdate('cascade')
        ->onDelete('cascade');
});
```

```php
public function down()
{
    Schema::dropIfExists('employees');
}
```

Ora che le tabelle sono state create, lanciando da terminale il comando *php artisan db:seed* verranno eseguiti in ordine i seeders dell'immagine sotto riportata che andranno a popolare le tabelle con dei dati da me preimpostati per eseguire dei test.



Conclusioni e miglioramenti

1. Assegnare un task ad uno sviluppatore
2. Rimuovere un task da uno sviluppatore

Per quanto riguarda i primi 2 punti, è stato previsto l'inserimento e la rimozione dei task nella pagina di dettaglio dell'utente. Sarebbe ulteriormente utile fornire la possibilità inversa ovvero, da una lista di task assegnare lo sviluppatore.

Per semplicità è stato previsto l'inserimento tramite l'ID del task. Sarebbe ulteriormente migliorabile, caricando in una SelectList, la lista di tutti i task per permettere all'utente di sceglierne uno (in base alla descrizione).

Inoltre, bisognerebbe prevedere tutti i controlli del caso: verificare se il task che si sta aggiungendo esiste veramente e che non appartenga già allo sviluppatore selezionato.

3. Mostrare tutti i task "in elaborazione" di uno sviluppatore
4. Mostrare i progetti cross-team (un progetto è cross team se ha sviluppatori di almeno 2 team diversi che lavorano ai suoi task)
5. *- Creare un nuovo DEV e assegnarlo ad un team

Per semplicità anche qui è stato previsto l'inserimento di un nuovo utente in base agli ID delle tabella correlate.
L'inserimento va a buon fine se l'ID dell'utente ha un ruolo CEO. Sarebbe migliorabile

prevedendo una schermata di login per gli account e, se l'account loggato è un utente con ruolo CEO allora avrà la possibilità di effettuare l'inserimento, in caso contrario no.

Per quanto riguarda gli altri parametri bisognerebbe controllare se essi esistono e non sono nulli.

6. Mostrare il PM di riferimento di un DEV

7. *- Mostrare i task che hanno sforato la deadline con i DEV che ci hanno lavorato e i loro relativi commits