IFTS 2025 - Project Work 2

Per questo esempio utilizziamo il modello dati dal testo del [Project Work 2](https://classroom.google.com/u/6/c/NzQ1OTM4MzE1MjIz/a/ODEwMzkxNTAxNDMz/details)
di Hotel e relative stanze:

- Proprietà per l'entità "Hotel": nome, immagine, indirizzo, numero di stelle,
  descrizione
- Proprietà per l'entità "Stanza": nome, immagine, tipologia, descrizione, prezzo

Backend Laravel

Seguendo le istruzioni dal [modulo di Laravel](https://github.com/docentemaurocasadei/laravel2025_api)

Creazione del progetto Laravel

```bash
# creazione della directory base
mkdir pw2-daniele-arduini
cd pw2-daniele-arduini

# creazione del progetto backend in Laravel
composer create-project laravel/laravel:"12.*" backend
cd backend
code .
```

Configurazione accesso DB

Con riferimento alla [documentazione Laravel per i parametri del DB](https://laravel.com/docs/11.x/database#configuration),
modificare i parametri di accesso al database all'interno del file `.env`.

Per conoscere i possibili driver per DB e relativi parametri di configurazione:

```bash
php artisan config:show database
```

Per impostare l'accesso al DB locale MariaDB:

- impostare i seguenti valori nel file `.env`:

  ```
  DB_CONNECTION=mariadb
  ```

```
  # DB_HOST=127.0.0.1
  # DB_PORT=3306
  DB_DATABASE=pw2-2025
  DB_USERNAME=root
  DB_PASSWORD=*******
  ```

- e rileggere i valori con:

  ```bash
  source .env
  ```

Creazione del modello dati

Con riferimento alla
[documentazione Laravel per le Database Migrations](https://
laravel.com/docs/12.x/migrations),
creare gli script per la creazione delle tabelle:

```bash
php artisan make:migration create_hotel_table --create=hotel
php artisan make:migration create_hotel_room_table --
create=hotel_room
```

Migrations per definire i campi del DB

Nel file `database/migrations/
2024_07_13_154116_create_hotel_table.php` si definiscono
i campi per la tabella `hotel` associata all'entità `Hotel`:

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('hotel', function (Blueprint $table) {
            $table->id();
            $table->string('nome');
            $table->string('immagine')->nullable()->default(null);
            $table->string('indirizzo')->nullable()-
>default(null);
```

```
            $table->string('stelle')->nullable()->default(null);
            $table->string('descrizione')->nullable()-
>default(null);
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('hotel');
    }
};
```

Nel file `database/migrations/2024_07_13_154126_create_hotel_room_table.php` si definiscono i campi per la tabella `hotel_room` associata all'entità `HotelRoom`:

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('hotel_room', function (Blueprint $table) {
            $table->id();
            $table->unsignedBigInteger('hotel_id');
            $table->string('nome');
            $table->string('immagine')->nullable()->default(null);
            $table->string('tipologia')->nullable()-
>default(null);
            $table->string('descrizione')->nullable()-
>default(null);
            $table->string('prezzo')->nullable()->default(null);
            $table->timestamps();

            $table->foreign('hotel_id')->references('id')-
>on('hotel')->onDelete('cascade');
        });
    }
```

```
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('hotel_room');
    }
};
```

Models per definire le entità in Laravel

Con riferimento alla
[documentazione Laravel per Eloquent ORM](https://laravel.com/docs/12.x/eloquent),
creare le entità per il modello dati:

```bash
php artisan make:model Hotel
php artisan make:model HotelRoom
```

Personalizzare il modello generato per `Hotel` in `app/Models/Hotel.php`:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Hotel extends Model
{
    use HasFactory;

    // personalizzazione del nome della tabella
    protected $table = 'hotel';

    // relazione 1 a molti con HotelRoom
    public function hotelRoomList(): HasMany {
        return $this->hasMany(HotelRoom::class, 'hotel_id');
    }
}
```

Personalizzare il modello generato per `HotelRoom` in `app/Models/HotelRoom.php`:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class HotelRoom extends Model
{
    use HasFactory;

    // personalizzazione del nome della tabella
    protected $table = 'hotel_room';

    // relazione con Hotel
    public function hotel(): BelongsTo {
        return $this->belongsTo(Hotel::class, 'hotel_id');
    }

}
```

Esecuzione delle migrations per applicare le modifiche al DB

```bash
php artisan migrate
```

Risposta:

```
php artisan migrate

   WARN  The database 'pw2-daniele-arduini-2024' does not exist on
the 'mariadb' connection.

  ┌ Would you like to create it? ─────────────────────────────────┐
  │ Yes                                                           │
  └───────────────────────────────────────────────────────────────┘

   INFO  Preparing database.

  Creating migration
table ..................................................... 35.67ms
DONE

   INFO  Running migrations.
```

```
0001_01_01_000000_create_users_table ..............................
.......... 234.28ms DONE

0001_01_01_000001_create_cache_table ..............................
.......... 53.58ms DONE

0001_01_01_000002_create_jobs_table ..............................
.......... 183.03ms DONE

2024_07_13_154116_create_hotel_table ..............................
.......... 23.17ms DONE

2024_07_13_154126_create_hotel_room_table .........................
.......... 73.65ms DONE
```

Inserimento dati con Seeder

Con riferimento alla
[documentazione Laravel per il database seeding](https://laravel.com/docs/11.x/seeding),
configurare il popolamento dei dati:

```bash
php artisan make:seeder HotelSeeder
```

in `database/seeders/DatabaseSeeder.php` aggiungere la chiamata a HotelSeeder:

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     */
    public function run(): void
    {
        $this->call([
            HotelSeeder::class,
        ]);
    }
}
```

in `database/seeders/HotelSeeder.php` aggiungere le insert per popolare il DB:

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class HotelSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        DB::table('hotel')->insert([
            'id' => 101,
            'nome' => 'Park Hotel',
            'immagine' => 'https://www.parkhotelrimini.com/wp-content/uploads/2019/01/park-hotel-rimini-1.jpg',
            'indirizzo' => 'Viale Regina Elena 6 - Rimini 47921 RN Emilia Romagna Italia',
            'stelle' => '4',
            'descrizione' => 'Il Park Hotel di Rimini sul mare ha una nuovissima gestione e tante novità. Uno staff gentile sempre a tua disposizione, tante occasioni per un soggiorno fronte mare, ...',
        ]);

            DB::table('hotel_room')->insert([
                'id' => 101,
                'hotel_id' => 101,
                'nome' => 'Camera Singola',
                'immagine' => 'https://www.riminiresort.it/wp-content/uploads/2023/12/heade-camera-singola-riminiresort.jpg',
                'tipologia' => 'Singola',
                'descrizione' => 'Le Camere singole sono ideali per chi viaggia solo/a e cerca la comodità al giusto prezzo. Ogni Camera singola è dotata un piacevole balcone. ...',
                'prezzo' => '70,00',
            ]);

            DB::table('hotel_room')->insert([
                'id' => 102,
                'hotel_id' => 101,
                'nome' => 'Camera Matrimoniale',
```

```
                'immagine' => 'https://www.riminiresort.it/wp-
content/uploads/2017/12/Slide-Camera-standard-03-Park-Hotel-
Rimini.jpg',
                'tipologia' => 'Matrimoniale',
                'descrizione' => 'Le Camere Matrimoniali ti
accolgono in un ambiente spazioso perfetto per chi in vacanza
cerca il massimo della qualità al giusto prezzo. ...',
                'prezzo' => '90,00',
            ]);



        DB::table('hotel')->insert([
            'id' => 201,
            'nome' => 'Hotel Polo',
            'immagine' => 'https://www.hotelpolo.it/inc/scripts/
source/www.hotelpolo.it/fil-zoom.png-9-crp630x510-facciata.jpg?
o=1',
            'indirizzo' => 'Viale A.Vespucci, 23, Marina Centro,
47900 Rimini, Italia',
            'stelle' => '4',
            'descrizione' => 'Immaginate di...arrivare a Rimini,
parcheggiare comodamente la vostra auto e pensare solo a cosa vi
fa’ stare bene. ...',
        ]);

            DB::table('hotel_room')->insert([
                'id' => 201,
                'hotel_id' => 201,
                'nome' => 'Camera Singola',
                'immagine' => 'https://www.riminiresort.it/wp-
content/uploads/2023/12/heade-camera-singola-riminiresort.jpg',
                'tipologia' => 'Singola',
                'descrizione' => 'Le Camere singole sono ideali
per chi viaggia solo/a e cerca la comodità al giusto prezzo. Ogni
Camera singola è dotata un piacevole balcone. ...',
                'prezzo' => '70,00',
            ]);

            DB::table('hotel_room')->insert([
                'id' => 202,
                'hotel_id' => 201,
                'nome' => 'Camera Matrimoniale',
                'immagine' => 'https://www.riminiresort.it/wp-
content/uploads/2017/12/Slide-Camera-standard-03-Park-Hotel-
Rimini.jpg',
                'tipologia' => 'Matrimoniale',
                'descrizione' => 'Le Camere Matrimoniali ti
accolgono in un ambiente spazioso perfetto per chi in vacanza
cerca il massimo della qualità al giusto prezzo. ...',
                'prezzo' => '90,00',
            ]);
```

```
    }
}
```

Eseguire il Seeder per popolare il DB:

```bash
➜  php artisan db:seed

   INFO  Seeding database.


Database\Seeders\HotelSeeder ..................................
............................ RUNNING

Database\Seeders\HotelSeeder ..................................
...................... 18.84 ms DONE
```

In caso di errori nell'importazione dati, come ad esempio elementi già presenti oppure dimensioni dei campi non sufficienti a contenere i dati da importare, correggere il problema ed utilizzare il seguente comando per rifare da zero tabelle e seed:

```bash
php artisan migrate:fresh --seed
```

Creazione dei Controller e delle Route

Creazione dei Controller

Con riferimento alla [documentazione Laravel per i Controller](https://laravel.com/docs/12.x/controllers), creare i controller per le 2 entità:

```bash
php artisan make:controller HotelController
php artisan make:controller HotelRoomController
```

in `app/Http/Controllers/HotelController.php`:

```php
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Hotel;

class HotelController extends Controller
```

```
{
    // GET http://localhost:8000/api/hotel/:id
    public function read(Request $request, $id)
    {
        $hotel = Hotel::where('id', '=', $id)-
>with('hotelRoomList')->firstOrFail();
        // $hotel = Hotel::where('id', '=', $id)->firstOrFail();
        return response()->json($hotel);
    }

    // GET http://localhost:8000/api/hotel
    public function readAll(Request $request)
    {
        // $hotel_list = Hotel::with('hotelRoomList')->get();
        $hotel_list = Hotel::get();
        return response()->json($hotel_list, 200);
    }
}
```

in `app/Http/Controllers/HotelRoomController.php`:

```php
<?php
namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\HotelRoom;

class HotelRoomController extends Controller
{
    // GET http://localhost:8000/api/hotel/:hotel_id/room/:room_id
    public function read(Request $request, $hotel_id, $room_id)
    {
        $hotel_room = HotelRoom::where([
            ['hotel_id', '=', $hotel_id],
            ['id', '=', $room_id]
        ])->firstOrFail();
        return response()->json($hotel_room);
    }

    // GET http://localhost:8000/api/hotel/:hotel_id/room
    public function readAll(Request $request, $hotel_id)
    {
        $hotel_room_list = HotelRoom::where('hotel_id', '=',
$hotel_id)->get();
        return response()->json($hotel_room_list, 200);
    }
}
```

Creazione delle Route

Con riferimento alla
[documentazione Laravel per le Route API](https://laravel.com/docs/12.x/routing#api-routes),
installare [Laravel Sanctum](https://laravel.com/docs/11.x/sanctum), che aggiunge a Laravel le funzionalità utili a realizzare API REST, eventualmente protette da autenticazione tramite token, che può essere utilizzata per autenticare i client di API di terze parti, SPA o applicazioni mobili.

```bash
php artisan install:api
```

Inoltre, il comando install:api crea il file `routes/api.php` che definisce le route per il
path `/api/...`, che ora va personalizzato aggiungendo le route per le nostre entità:

```php
use App\Http\Controllers\HotelController;
use App\Http\Controllers\HotelRoomController;

Route::middleware('api')->group(function () {
    // GET http://localhost:8000/api/hotel/:hotel_id
    Route::get('/hotel/{hotel_id}', [HotelController::class, 'read']);

    // GET http://localhost:8000/api/hotel
    Route::get('/hotel', [HotelController::class, 'readAll']);

    // GET http://localhost:8000/api/hotel/:hotel_id/room/:room_id
    Route::get('/hotel/{hotel_id}/room/{room_id}', [HotelRoomController::class, 'read']);

    // GET http://localhost:8000/api/hotel/:hotel_id/room
    Route::get('/hotel/{hotel_id}/room', [HotelRoomController::class, 'readAll']);
});
```

Verifica delle Route

```bash
php artisan route:list

  GET|HEAD        / ..........................................................
  GET|HEAD        api/hotel ..................................................
HotelController@readAll
```

```
  GET|HEAD       api/hotel/
{hotel_id} ..............................................
HotelController@read
  GET|HEAD       api/hotel/{hotel_id}/
room ....................................
HotelRoomController@readAll
  GET|HEAD       api/hotel/{hotel_id}/room/
{room_id} ............................ HotelRoomController@read
  GET|HEAD       api/
user ..........................................................
...................
  GET|HEAD       sanctum/csrf-cookie .. sanctum.csrf-cookie ›
Laravel\Sanctum › CsrfCookieController@show
  GET|HEAD
up ............................................................
.......................

Showing [8] routes
```

Avvio del backend Laravel

```bash
php artisan serve
```

Test del backend

Per verificare il corretto funzionamento del backend e verificare
i campi restituiti nel JSON delle
risposte alle chiamate REST, è possibile utilizzare Postman oppure
un semplice browser web visto
che nel nostro caso ci sono solamente chiamate GET

Per i test, richiamare i seguenti URL dal browser:

- http://localhost:8000/api/hotel
- http://localhost:8000/api/hotel/101
- http://localhost:8000/api/hotel/101/room
- http://localhost:8000/api/hotel/101/room/101
- ...

Frontend Angular

Creazione del progetto Angular

```bash
cd pw2-daniele-arduini

# creazione del progetto frontend in Angular
```

```
ng new frontend --skip-tests --style css --standalone false --ssr
false

cd frontend
code .
```

Aggiunta libreria Bootstrap

```bash
➜  npm install bootstrap --save
```

in `src/styles.css`:

```css
/* You can add global styles to this file, and also import other
style files */

@import "bootstrap/dist/css/bootstrap.css";
```

Aggiunta libreria FontAwesome

```bash
➜  npm install --save @fortawesome/fontawesome-free
```

in `src/styles.css`:

```css
/* You can add global styles to this file, and also import other
style files */

@import "@fortawesome/fontawesome-free/css/all.css";
```

Creazione del modello dati

creare il file `src/app/hotel.model.ts` e creare le classi (o
interfacce) del modello dati della API REST:

```typescript
export interface Hotel {
  id: number;
  nome: string;
  immagine: string;
  indirizzo: string;
  stelle: string;
  descrizione: string;

  hotel_room_list: HotelRoom[];
```

```
}

export interface HotelRoom {
  id: number;
  hotel_id: number;
  nome: string;
  immagine: string;
  tipologia: string;
  descrizione: string;
  prezzo: string;
}
```

Creazione del Service

Importare HttpClientModule

in `src/app/app.module.ts`:

```typescript
...
import { HttpClientModule } from '@angular/common/http';
...

@NgModule({
  ...
  imports: [
    ...
    HttpClientModule,
    ...
  ],
  ...
})
export class AppModule { }
```

Creare l'HotelService

```bash
➜  ng generate service hotel --skip-tests
CREATE src/app/hotel.service.ts (134 bytes)
```

in `src/app/hotel.service.ts`:

```typescript
import { Injectable } from "@angular/core";
import { HttpClient } from "@angular/common/http";
import { Hotel, HotelRoom } from "./hotel.model";

@Injectable({
  providedIn: "root",
```

```
})
export class HotelService {
  baseUrl = "http://localhost:8000";

  constructor(private http: HttpClient) {}

  getHotelList(): Promise<Hotel[] | undefined> {
    return this.http.get<Hotel[]>(`${this.baseUrl}/api/
hotel`).toPromise();
  }

  getHotel(id: number): Promise<Hotel | undefined> {
    return this.http.get<Hotel>(`${this.baseUrl}/api/hotel/${id}
`).toPromise();
  }

  getHotelRoomList(hotelId: number): Promise<HotelRoom[] |
undefined> {
    return this.http
      .get<HotelRoom[]>(`${this.baseUrl}/api/hotel/${hotelId}/
room`)
      .toPromise();
  }

  getHotelRoom(
    hotelId: number,
    roomId: number
  ): Promise<HotelRoom | undefined> {
    return this.http
      .get<HotelRoom>(`${this.baseUrl}/api/hotel/${hotelId}/room/$
{roomId}`)
      .toPromise();
  }
}
```

Creazione dei Component

Creazione HotelRoomComponent

```bash
➜  ng generate component hotel-room --skip-tests --inline-style
CREATE src/app/hotel-room/hotel-room.component.html (25 bytes)
CREATE src/app/hotel-room/hotel-room.component.ts (189 bytes)
UPDATE src/app/app.module.ts (567 bytes)
```

Creazione HotelComponent

```bash
➜  ng generate component hotel --skip-tests --inline-style
CREATE src/app/hotel/hotel.component.html (20 bytes)
```

```
CREATE src/app/hotel/hotel.component.ts (175 bytes)
UPDATE src/app/app.module.ts (471 bytes)
```