Peppero Locks

build passing

tl;dr

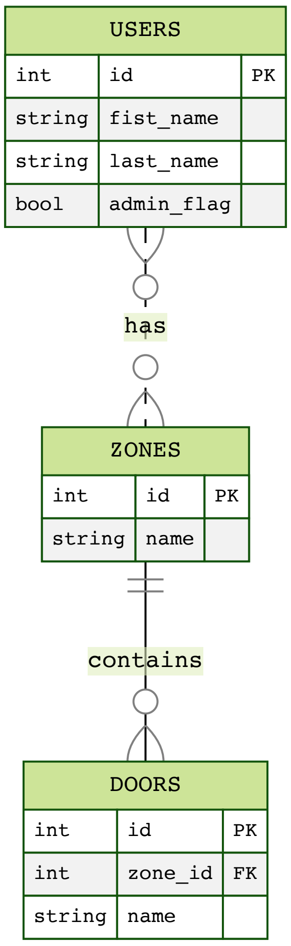
This README details the model relations and functionality included in *Peppero-Locks*.

All class files are completed and adequately commented, so this document is purely supplementary.

Screenshots from the `php artisan tinker` CLI can be seen in the final section of this readme.

The full project repository can be found on GitHub.

Entity Relationship Diagram



Explicit access

Access can be provided to a user by adding a record to the `direct_access` table. As this is a many-to-many relationship (many users could have direct access to the same door), an additional `direct_access` table is used to store these associations (see create_doors_users_table).

```
/** Check which doors the first user has access to */
$user = User::first();
$user->doors;

/** Add explicit access to the first door */
$user->doors()->attach(Door::first());

/** Or using the provided method */
$user->addAccessToDoor(Door::first());
```

Zonal Access

Once again, the relationship between Doors and Zones is a many-to-many relationship, so an additional *pivot* table, `zonal_access` stores these relations (see create_users_zones_table).

The database has been seeded with 10 zones, 1 of which contains no doors (Zone 1) and one of which has no zonal access given to any users (Zone 10).

```
/** Get a collection of doors assigned to Zone 1 */
Zone::find(1)->doors

/** Identify all users which have access to Zone 10 */
Zone::find(10)->users
```

User::where('id', 2)->get() Zone::find(1)->doors Door::first()->zone->users

Admin access

Admin users have access to all doors implicitly, which can be shown by supplying any Door to the `hasAccessToDoor()` method (see below).

This does not mean that all doors wil be returned by the `doors` and `zoneDoors` properties (which return explicitly accessible doors and zonally accessible doors respectively).

```
/** Check if user with id 5 is an admin */
User::find(5)->isAdmin()

/** Update user with id 6 to be an admin */
User::query()->where('id', 5)->update(['admin_flag' => true]);
```

`hasAccessToDoor()`

This method indicates whether a user has access to a supplied door, through either:

- Direct access (`direct_access` table) → verified via the `doors` property in User.
- Zone access (`zonal_access` table) → verified via the `zoneDoors` property in User.
- Admin rights (`users.admin_flag == true`) → checked using the `isAdmin()` method in User.

```
/** Update user with id 6 to be an admin */
User::find(1)->hasAccessToDoor(Door::find(616))

/** Update user with id 6 to be an admin */
User::find(1)->hasAccessToDoor(Door::find(711))

/** Update user with id 6 to be an admin */
User::firstWhere('admin_flag', false)->zoneDoors
```

User expiry

All users must include a value for `expiry_date` , after which point they will not be able to access any doors, regardless of their admin status.

```
/** Check if the first User is active */
User::first()->isActive()

/** Manually expire a User */
/** Manually expire a user */
User::find(3)->expire()
```

Transitive relationships (`HasManyThrough`)

The doors that can be accessed by a given user through *zone*-access can be listed using the `zoneDoors` property in User.

Except for the many-to-amy relationship between zones and users, this would be accessed using a `HasManyThrough` Eloquent relationship, but because of the many-to-many relationship, there is instead a manual implementation.

```
/** Get doors which are explicitly accessible for user with id 2 */
User::find(2)->doors

/** Get doors from all zones which user with id 2 has access to */
User::find(2)->zoneDoors
```

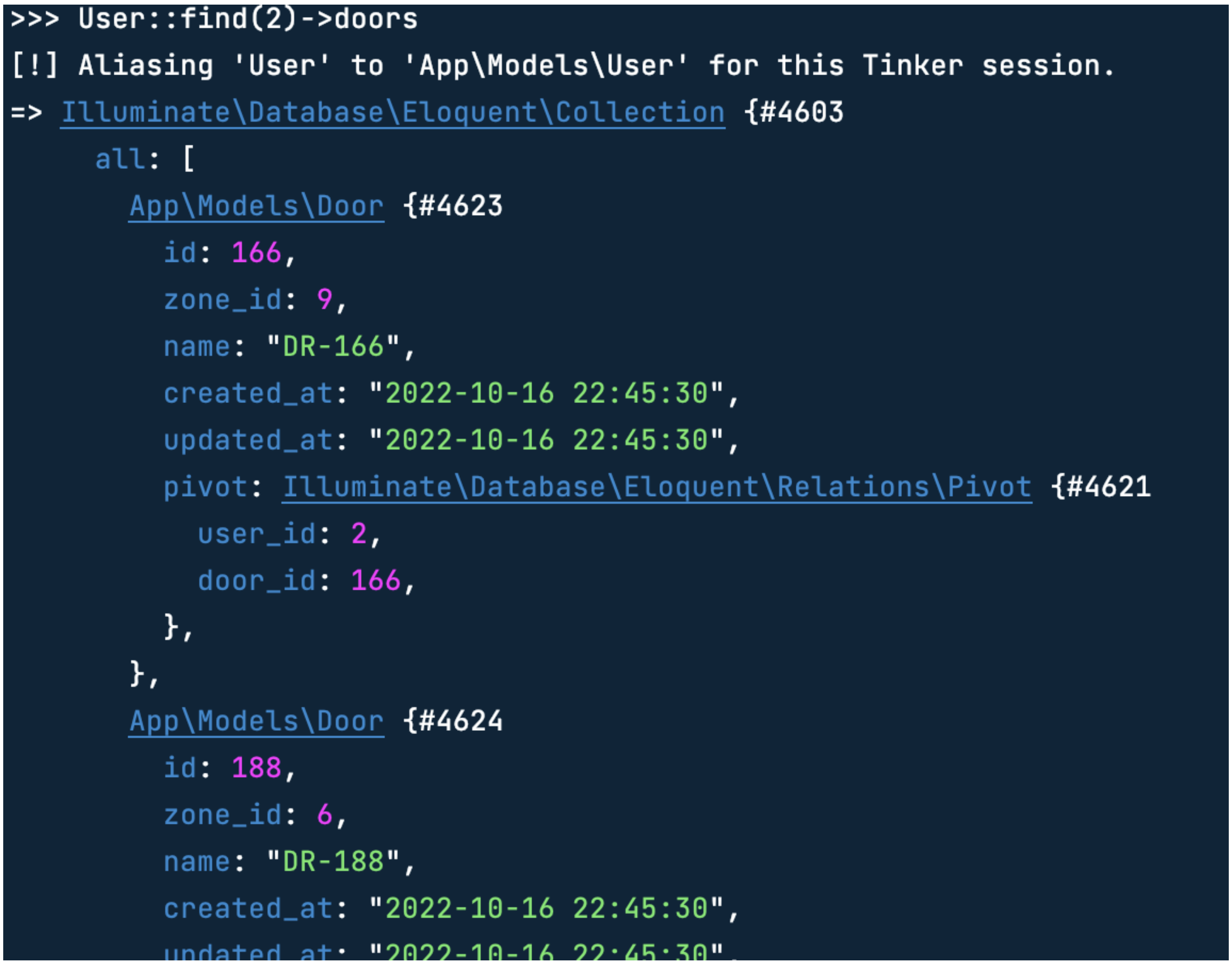
This latter property would be similar to a SQL query like:

```
SELECT
     a.user_id
    ,d.zone_id
    ,d.id AS door_id
    ,d.name
FROM doors d
INNER JOIN zonal_access a
    ON d.zone_id = a.zone_id
WHERE a.user_id = 2
ORDER BY zone_id, door_id;
```

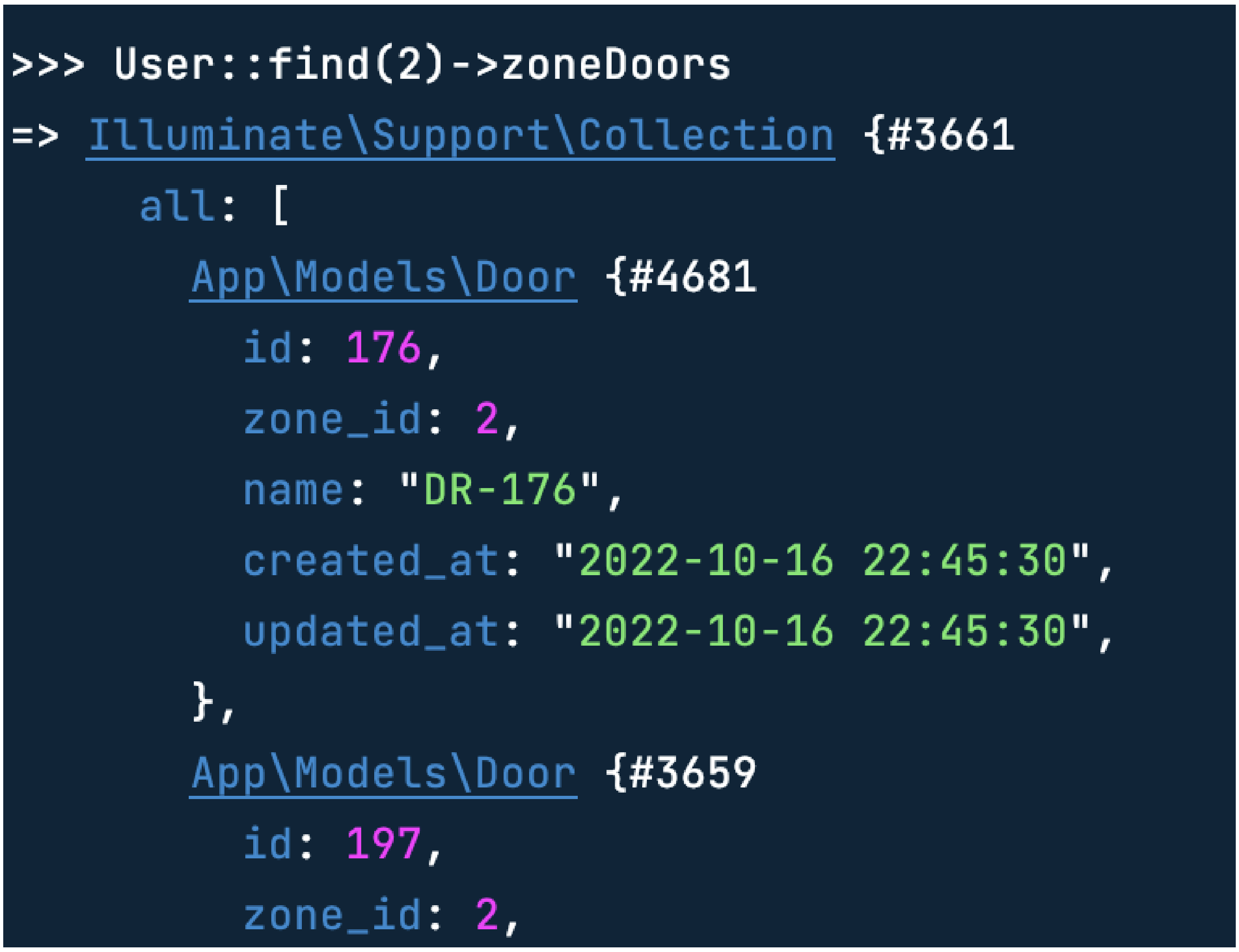
References

PHPDocs and other conventions have been taken from IxFD.

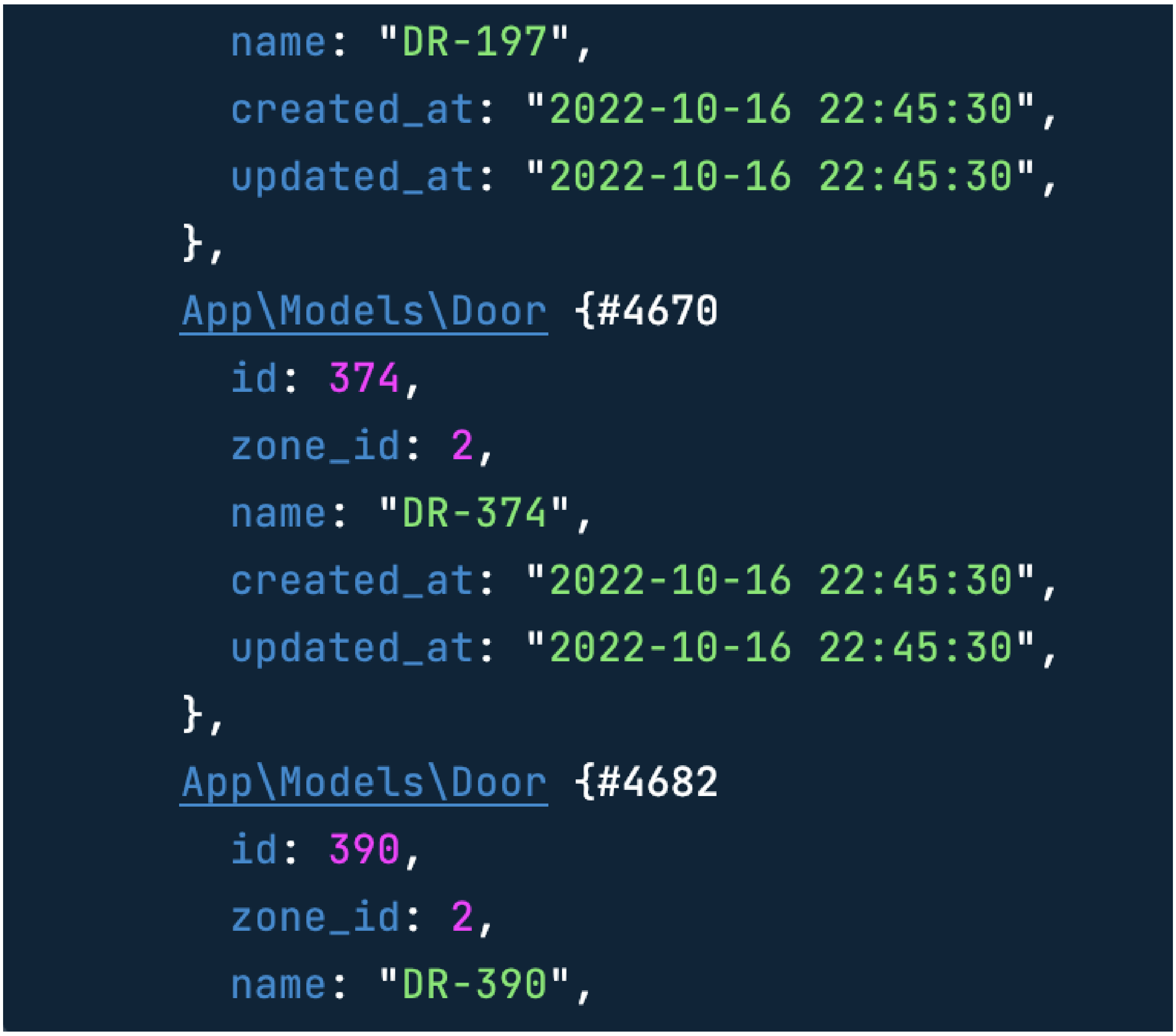
Tinker screenshots



```
>>> User::find(2)->doors
[!] Aliasing 'User' to 'App\Models\User' for this Tinker session.
=> Illuminate\Database\Eloquent\Collection {#4603
     all: [
       App\Models\Door {#4623
         id: 166,
         zone_id: 9,
         name: "DR-166",
         created_at: "2022-10-16 22:45:30",
         updated_at: "2022-10-16 22:45:30",
         pivot: Illuminate\Database\Eloquent\Relations\Pivot {#4621
           user_id: 2,
           door_id: 166,
         },
       },
       App\Models\Door {#4624
         id: 188,
         zone_id: 6,
         name: "DR-188",
         created_at: "2022-10-16 22:45:30",
         updated_at: "2022-10-16 22:45:30",
```



```
>>> User::find(2)->zoneDoors
=> Illuminate\Support\Collection {#3661
     all: [
       App\Models\Door {#4681
         id: 176,
         zone_id: 2,
         name: "DR-176",
         created_at: "2022-10-16 22:45:30",
         updated_at: "2022-10-16 22:45:30",
       },
       App\Models\Door {#3659
         id: 197,
         zone_id: 2,
```



```
        name: "DR-197",
        created_at: "2022-10-16 22:45:30",
        updated_at: "2022-10-16 22:45:30",
      },
      App\Models\Door {#4670
        id: 374,
        zone_id: 2,
        name: "DR-374",
        created_at: "2022-10-16 22:45:30",
        updated_at: "2022-10-16 22:45:30",
      },
      App\Models\Door {#4682
        id: 390,
        zone_id: 2,
        name: "DR-390",
```