Hacettepe University
Computer Science and Engineering Department

Name and Surname : Emin Orçun Sancar & Musa ALİŞAR
Identity Number : 21427306 & 21426567
Course : BBM 473 Data Management Systems Laboratory
Experiment : Project Phase -2-
Subject : Online Food Ordering System Final Project
Data Due : 13.01.2019
Advisors : Research Teaching Assistant Aysun KOÇAK

Project Definition

We made Online Food Ordering System web application. This system is similar to Yemek Sepeti.com . We have 4 main user types.

-User(Customer)

-Manager

-Admin

-Restaurant Admin

It is easy to use as a web application.Each table can defined in the system and added successfully.

These user types have their own unique works.

Admin Specific Scenario and Functions

Admin have some specific works. This user type can do evething in the system.So admin is the top user in inheritance hierarchy. For example A admin can check Restaurants and also do some CRUD operations on restaurant. They can make CRUD operations on Restaurant products also,Admin can add,delete or update users, products and orders. Admin can see commands and see scoreboard. Like every user admin can update password and informations about his/her profile.

For Example:We have a admin and his name is Ahmet. Ahmet can add Restaurant that name is “Ciğerci Apo”, he can add some product the this restaurant For example he add Ciğer kavurma which price is 20 tl and type which is Ciğer then he can update or delete these specific Restaurant and its food according to his own choice.Ahmet can see orders according to users choices and see scoreboard for example users Veli, Furkan and Musa and see their comments about restaurant and order. Admin can update his/her profile.

Restaurant Admin Specific Scenario and Functions

Restaurant Admin have some specific works. This user type can control Restaurants in the system for example A restaurant admin can check Restaurants and also do some CRUD operations on restaurant. They can make CRUD operations on Restaurant products also.Restaurant admins can add delete or update products of his/her restaurant. Like every user restaurant admin can update password and informations about his/her profile.

For Example:We have a restaurant admin and his name is Sami. Sami can add Restaurant that name is “Ciğerci Apo”, he can add some product the this restaurant For example he add Ciğer kavurma which price is 20 tl and type which is Ciğer then he can update or delete these specific Restaurant and its food according to his own choice. Restaurant Admin can update his/her profile.

Manager Specific Scenario and Functions

Manager have some specific works. This user type can do payment transformantions about users and restaurants and check their budgets. For example A manager can check Restaurants and Users and observe depth of them and send them payment form according to their depths. They can make CRUD operations on Payment also. Like every user manager can update password and informations about his/her profile.

For Example:We have a manager and his name is Mehmet. Mehmet can check Restaurant that name is “Ciğerci Apo”,and its depth, Mehmet can check user that name is “Orçun”,and its depth, and send them a payment screen and update payment table according to payments. Manager can update his/her profile.

User(Customer) Specific Scenario and Functions

User have some specific works. User can do every operation about his/her profile and orders. For example A user can update his profile and order products according to choice.User can see his recent movements according to orders They can make CRUD operations on orders also. Like every user customer can update password and informations about his/her profile

For Example:We have a user and his name is Ali. Ali can add order from "Ciğerci Apo", and he can choose ciğer kavurma.If user change his mind he/she can update or delete his order or he/she can order from different restaurant.User can update his/her profile.

Required items Database Design

We have some required items in the system:
In database side, Tables defined in Context in the below and we used phpmyadmin for database connection we configure our localhost,database name,database user name and password then make connection in phpmyadmin.
In web application side we used laravel and we designed our project by using some methods:
-We defiene each table in database/migrations
-Secondly we create model each table in App/ and we define relations and foreign key constraints in there
-We create controller for each model in App/Http/Controllers and define functions that we use in the system.
-We create authantication for login system in App/Http/Controllers/Auth and define register and login cotroller. Also we define forgetPassword and resetPassword in our system.
-We create RedirectedIfAuthenticated in App/Http/Controllers/Middleware authenticate for distinct user types.
-We insert inside auth.php in /app/config for distinct user types.
-We insert app.blade in /resources/views for dynamic view.

-Lastly we create page of each distinct users and our model in the resources/views.
For example order model we create our files in: resources/views/order and add 3 pages for insert delete update and show operations in the view.
-Lastly we route our pages in routes/web.php according to our design.

Relations in Database Design

We have 4 main user types:
-User(Customer)

-Manager

-Admin

-Restaurant Admin

And other tables are;

-Address:address id is the foreign key of all user types. And each users address can find from Address table

- CustomerPayment:it takes customer id as foreign key and each users informations come from User table

- CustomerMembership: it takes customer id as foreign key and each users informations come from User table

- RestaurantPayment: it takes restaurant id as foreign key and each users informations come from Restaurant table

- RestaurantMembership: it takes restaurant id as foreign key and each users informations come from Restaurant table

- MyFavourites:it takes product id as foreign key and we take favourites from product table

- RecentMovements:It take user id as foreign key and select last operations from user table

- Scoreboard:It takes order id as foreign key and select score of each order from orders table.

- Comments:It takes restaurant id as foreign key and select comments from each restaurant

- Snack:It takes product id as foreign key and select snacks from each product

- Menu:It takes product id as foreign key and select menus from each product

- Product

- Order:It takes product id as foreign key and select product name from product table.

- Restaurant

For example, lets look at our Product-Order Relation

Each order has one product and each product may have been many orders.So we add product id as a foreign key in Order.

We create our table in database/migrations.

Definition of order table is below and add foreign key as a normal variable.

_13_001605_create_orders_table.php

```php
     * @return void
     */
    public function up()
    {
        Schema::create( table: 'orders', function (Blueprint $table) {
            $table->increments( column: 'id');
            $table->string( column: 'name');
            $table->string( column: 'phone');
            $table->text( column: 'address');
            $table->date( column: 'delivery_date');
            $table->integer( column: 'product_id');
            $table->string( column: 'payment_option');
            $table->integer( column: 'quantity');
            $table->string( column: 'order_status')->nullable();
            $table->nullableTimestamps();
        });
    }
```

Definition of order table is below.

01_13_001623_create_products_table.php

```php
class CreateProductsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create( table: 'products', function (Blueprint $table) {
            $table->increments( column: 'id');
            $table->string( column: 'product_name');
            $table->string( column: 'product_type');
            $table->text( column: 'product_details');
            $table->integer( column: 'price');
            $table->timestamps();
        });
    }
```

We create models in app/.

Definition of order model is below.

Each order belongs to Product.

Order.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Order extends Model
{
    public function product(){
        return $this->belongsTo( related: 'App\Product');
    }
}
```

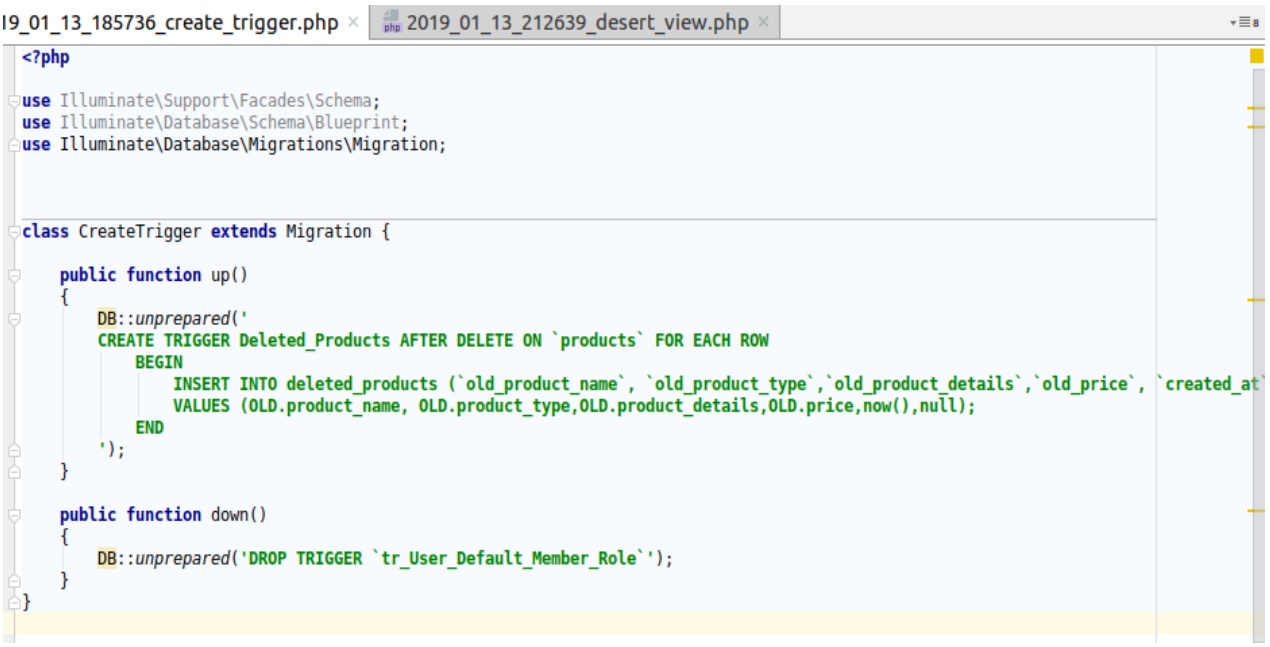
Definition of product model is below.

Each order has many relation with product.

Product.php

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Product extends Model
{
    public function orders(){
        return $this->hasMany( related: 'App\Order', foreignKey: 'product_id');
    }
```

We use trigger for detect deleted products:



We use transaction for add users and attributes (values) in our tables by interleavly (paralelly) appropriate serializable:

```php
DB::beginTransaction();

try {

    DB::table('admins')->insert(
        ['id' => '1', 'name' => 'admin1', 'email' => 'admin1@gmail.com',
            'password' => '$2y$10$6idzub6jqP2L40VJy0rTJ.Dx/GMlndQtjQfQY1zpZIGvpsRmrGiAK']
    );
    DB::table('restaurantadmins')->insert(
        ['id' => '1', 'name' => 'restaurantadmin1', 'email' => 'restaurantadmin1@gmail.com',
            'password' => '$2y$10$6idzub6jqP2L40VJy0rTJ.Dx/GMlndQtjQfQY1zpZIGvpsRmrGiAK']
    );
    DB::table('managers')->insert(
        ['id' => '1', 'name' => 'manager1', 'email' => 'manager1@gmail.com',
            'password' => '$2y$10$6idzub6jqP2L40VJy0rTJ.Dx/GMlndQtjQfQY1zpZIGvpsRmrGiAK']
    );
    DB::table('users')->insert(
        ['id' => '1', 'name' => 'user1', 'email' => 'user1@gmail.com',
            'password' => '$2y$10$6idzub6jqP2L40VJy0rTJ.Dx/GMlndQtjQfQY1zpZIGvpsRmrGiAK']
    );
    DB::table('products')->insert(
        ['id' => '1', 'product_name' => 'Tavuk Durum', 'product_type' => 'Fast Food',
            'product_details' => 'Tavuk Durum', 'price' => 9]
    );
    DB::table('orders')->insert(
        ['id' => '1', 'name' => 'Orcun', 'phone' => '3333333333',
            'address' => 'Beytepe', 'delivery_date' => '2019-01-14',
            'product_id' => '1', 'payment_option' => 'Postpay',
            'quantity' => '3', 'order_status' => 'Confirm']
```

php × 2019_01_12_202254_create_restaurants_table.php ×

```
            'product_details' => 'Kaşarlı Kavurmalı Pide', 'price' => 10]
    );
    DB::table('users')->insert(
        ['id' => '10', 'name' => 'user10', 'email' => 'user10@gmail.com',
            'password' => '$2y$10$6idzub6jqP2L40VJy0rTJ.Dx/GMlndQtjQfQY1zpZIGvpsRmrGiAK']
    );
    DB::table('restaurants')->insert(
        ['id' => '4', 'name' => 'Aslı Börek', 'account' => 1000,
            'city' => 'Ankara', 'district' => 'Beytepe',
            'phone' => '054352452532', 'product_id' => '6']
    );
    DB::table('products')->insert(
        ['id' => '10', 'product_name' => 'Mantı', 'product_type' => 'Ana Yemek',
            'product_details' => 'Mantı', 'price' => 15]
    );
    DB::table('restaurants')->insert(
        ['id' => '3', 'name' => 'Fesleğen', 'account' => 1100,
            'city' => 'Ankara', 'district' => 'Beytepe',
            'phone' => '053423141', 'product_id' => '5']
    );


    DB::commit();

} catch (\Exception $e) {
    DB::rollback();

}
```

We use views for create various helpful virtual tables:

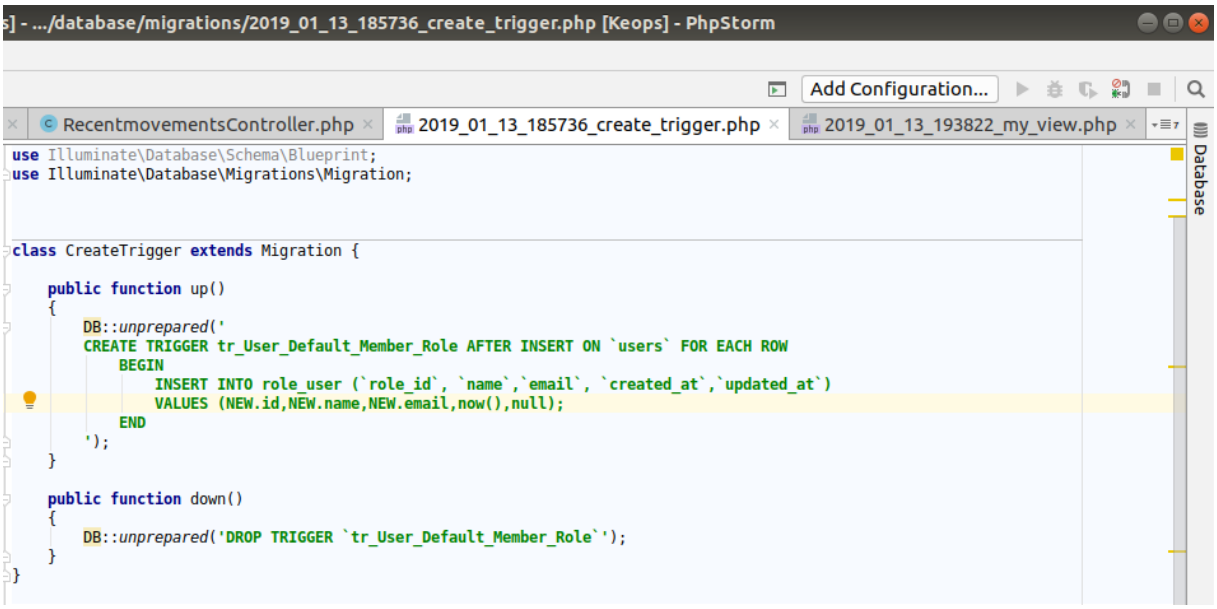
```
<?php

use ...

class StartupRestaurantView extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        DB::statement( 'CREATE VIEW startUpRestaurantView AS SELECT name FROM restaurants WHERE account<=1000' );
    }

    public function down()
    {
        DB::statement( 'DROP VIEW startUpRestaurantView' );
    }
}
```

eb.php × 2019_01_13_204441_fast_food_view.php × 2019_01_12_202254_create_restaurants_table.php ×

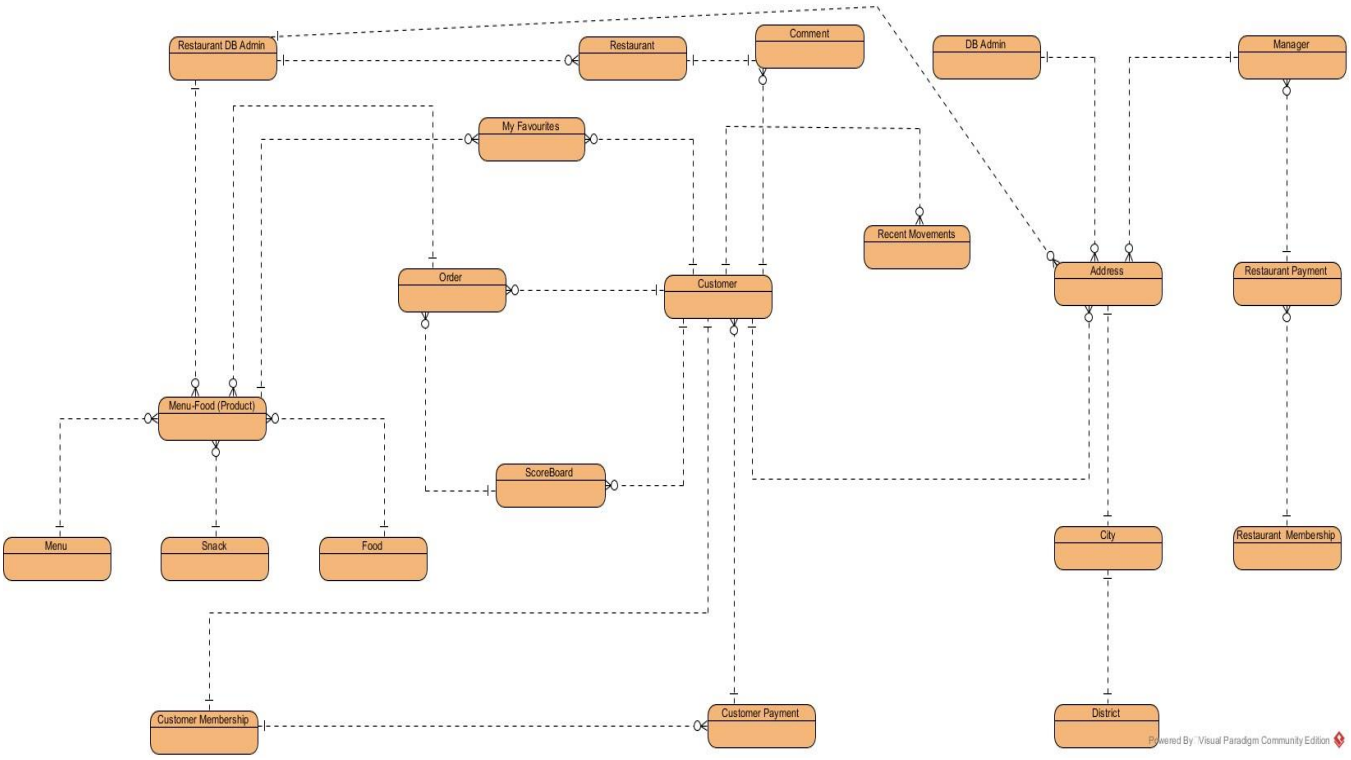
```php
<?php

use ...

class FastFoodView extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        DB::statement( 'CREATE VIEW fastFoodView AS SELECT product_name FROM products WHERE product_type="Fast Food" ' );
    }

    public function down()
    {
        DB::statement( 'DROP VIEW fastFoodView' );
    }
}
```

```php
<?php

use ...

class DesertView extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        DB::statement( 'CREATE VIEW desertView AS SELECT product_name FROM products WHERE product_type="Tatlı" ' );
    }

    public function down()
    {
        DB::statement( 'DROP VIEW desertView' );
    }
}
```

```php
<?php

use ...

class RichRestaurantView extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        DB::statement( 'CREATE VIEW richRestaurantView AS SELECT name FROM restaurants WHERE account>1000' );
    }

    public function down()
    {
        DB::statement( 'DROP VIEW richRestaurantView' );
    }
}
```



ER Diagram


Restaurant DB Admin
Restaurant
Comment
DB Admin
Manager
My Favourites
Recent Movements
Order
Customer
Address
Restaurant Payment
Menu-Food (Product)
ScoreBoard
Menu
Snack
Food
City
Restaurant Membership
Customer Membership
Customer Payment
District
Powered By Visual Paradigm Community Edition