# PROYECTO DE CGIS: APLICACIÓN WEB PARA UNA FARMACIA

PRIMERA ENTREGA: 8 MARZO 2018

MARÍA MILLÁN GAMERO

3º INGENIERÍA DE LA SALUD

CODIFICACIÓN Y GESTIÓN DE LA INFORMACIÓN SANITARIA

Usuario GitHub: mariamillan97

# 1. INTRODUCCIÓN

En las empresas farmacéuticas existen problemas de gestión de ventas, debido al mal seguimiento de éstas, dando lugar a la falta de productos para nuestros clientes, deudas no pagadas y ventas de fármacos sin receta que la necesitan para ser vendidos.

Este proyecto va destinado a facilitar la gestión de las ventas de las empresas farmacéuticas. El objetivo es crear una aplicación web que permita llevar un control de las ventas que se realizan en ésta y en la que los clientes de la farmacia puedan participar.

Las farmacias necesitan llevar una buena gestión de las ventas de los productos ya que llevando un buen control de éstas:

- ✓ Podrán saber cuál ha sido el consumo de cada producto en cualquier día del año.
- ✓ Conocer que productos han sido los más comprados por nuestros clientes.
- ✓ Saber si existe la necesidad de comprar más cantidad de un producto determinado.
- ✓ Controlar las deudas de nuestros clientes.

Los usuarios tendrán acceso a la aplicación web como clientes o como trabajadores.

Un trabajador podrá:

- Acceder a la app web con su usuario y contraseña.
- Saber el número de ventas realizadas en un día, en el mes y en el año; e incluso en una fecha determinada que se desee consultar.
- Acceder a toda la información almacenada de cada producto (código, nombre, tipo, precio de compra y precio de venta al público, suministradora, si necesita receta o no…).
- Podrá saber a qué cliente se le ha realizado una determinada venta.

Un cliente podrá:

- Acceder a la app web con su usuario y contraseña.
- Ver si tiene alguna deuda de alguna venta que no pagó (cantidad de la deuda, si no tiene aparece 0 euros).
- Tener acceso a sus datos y modificarlos: nombre, apellidos, DNI, email.
- Acceder al historial de los productos que ha comprado en dicha farmacia.
- Reservar productos para recogerlos en la farmacia.

Los beneficios que se obtendrán tras esta aplicación en las empresas farmacéuticas:

- Disminución de compras por impulso.
- Mayor número de existencias de algunos productos según lo que nuestros clientes compren.
- Tener un producto determinado si un paciente lo ha reservado.
- Disminución del tiempo de compra.
- Incremento del volumen de compra a su gran gestión.

## 2. REQUISITOS DE INFORMACIÓN

Los requisitos de información describen qué información se debe almacenar sobre un concepto relevante para poder cumplir los objetivos, y que datos específicos del concepto son importantes para los usuarios.

Los requisitos de información para llevar a cabo nuestro proyecto son los siguientes:

**RI-001 Información sobre los productos**

**Como** usuario
**Quiero** almacenar la información correspondiente a cada producto
En concreto: el código, el nombre, el tipo, precio de venta y de compra, fecha de caducidad, número de existencias, su proveedor y si necesita o no receta para poder ser vendido
**Para** poder localizarlo

**RI-002 Información sobre las ventas de productos**

**Como** usuario
**Quiero** almacenar todas las ventas de productos que se producen en un día. En concreto: la fecha de la venta, producto o productos vendidos, cantidad de esos productos, el precio total (cuenta), si es pagada o no y cómo, el cliente que compra y el trabajador que realiza la venta
**Para** saber que productos se han vendido, a qué precio y cuáles necesitaremos pedir.

**RI-003 Información sobre trabajadores de la farmacia**

**Como** usuario
**Quiero** disponer de la siguiente información sobre los empleados.
En concreto: nombre, apellidos, DNI, email, salario y el rol que tiene en dicha farmacia
**Para** poder gestionar su información y actualizarla si quisiera.

**RI-004 Información sobre los clientes**

**Como** usuario
**Quiero** disponer de la siguiente información sobre los clientes.
En concreto: nombre, apellidos, DNI, email, si tiene deudas o no, número de la Seguridad Social y los productos que ha comprado
**Para** satisfacer mejor las necesidades.

**RI-005 Información sobre los proveedores**

**Como** usuario
**Quiero** almacenar información sobre los proveedores de nuestros productos.
En concreto: nombre, dirección, email y tipo de productos que suministran
**Para** llevar una mejor gestión de las empresas a las que les realizan los pedidos y tomar las mejores decisiones de compra.

## 3. REQUISITOS FUNCIONALES

En general, definen los servicios que los usuarios desean que el sistema les ofrezca:

**RF-001 Visualización de ventas diarias**

**Como** trabajador
**Quiero** conocer las ventas realizadas en un día determinado
**Para** saber cuál ha sido el beneficio obtenido a lo largo del día

**RF-002 Información sobre lo que gasta cada cliente**

**Como** trabajador
**Quiero** consultar la media de lo que se gasta cada cliente
**Para** satisfacer mejor las necesidades de los clientes y tener un control de deudas.

**RF-003 Información sobre la relación entre los trabajadores, clientes y ventas**

**Como** trabajador
**Quiero** consultar las ventas y el trabajador y cliente que participan en dicha venta
**Para** poder gestionar el personal de la farmacia.

**RF-004 Consultar máximo y mínimo precio venta de cada trabajador**

**Como** trabajador
**Quiero** conocer el precio de la venta máxima y mínima que ha realizado dicho trabajador
**Para** tener mayor control de las ventas que atiende cada trabajador.

**RF-005 Consultar que productos vende cada trabajador**

**Como** trabajador
**Quiero** consultar que productos ha vendido cada trabajador en un día determinado
**Para** controlar las ventas.

**RF-006 Información sobre los productos que venden**

**Como** trabajador
**Quiero** consultar la información de los productos que les ofrezco a mis clientes. En concreto: el código, el nombre, el tipo, precio de venta y de compra, fecha de caducidad, número de existencias, su proveedor y si necesita o no receta para poder ser vendido
**Para** conocer mejor los productos que se van a vender.

**RF-007 Información sobre los productos que compran**

**Como** cliente
**Quiero** consultar la información de los productos que puedo reservar y comprar en la farmacia. En concreto: nombre, precio y si necesita receta o no
**Para** poder reservar el producto que desee.

**RF-008 Consultar historial de productos comprados**

**Como** cliente
**Quiero** consultar el historial de los productos que he comprado
**Para** llevar un control de los productos que debe comprar.

**RF-009 Información sobre las deudas totales**

**Como** trabajador
**Quiero** consultar que clientes tienen deudas y qué cantidad, pudiendo así acceder a sus datos. En concreto: nombre, apellidos, email y ventas realizadas.
**Para** llevar un control.

**RF-010 Consultar las deudas clientes**
**Como** cliente
**Quiero** saber si tengo alguna deuda con la farmacia
**Para** pagarla cuanto antes.

**RF-011 Reservar productos**

**Como** cliente
**Quiero** reservar productos que estén disponibles en la farmacia
**Para** poder agilizar el tiempo de compra.

**RF-012 Acceso a la app**

**Como** usuario
**Quiero** acceder a una determinada información según si soy un trabajador o un cliente
**Para** poder tener información de lo que me interesa.

## 4. REGLAS DE NEGOCIO

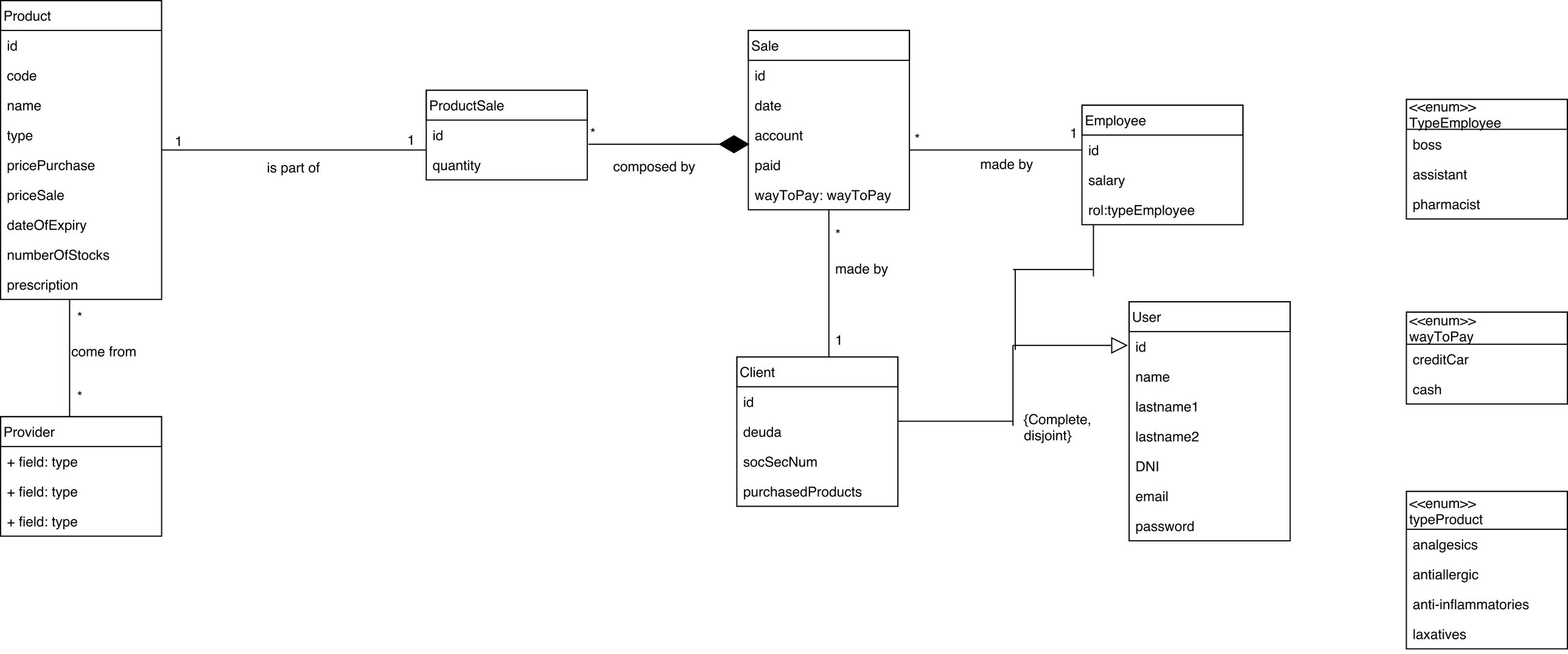
Las reglas de negocio definen reglas o políticas del negocio que son importantes para los usuarios y deben ser respectadas. Son requisitos que podrían cambiar en el futuro por cambios en la farmacia.

**RN-001 No borrar a trabajadores con sueldo superior a 3000 euros**

**Como** sistema
**Quiero** que no podamos eliminar a un trabajador con un sueldo mayor a 3000 euros
**Para** no borrar a los trabajadores que son fijos.

**RN-002 No eliminar clientes con pagos pendientes**

**Como** trabajador
**Quiero** evitar que se puedan eliminar los datos correspondientes a un cliente mientras éste tenga registradas facturas pendientes de pago
**Para** que los pagos se cumplan correctamente.


Product
id
code
name
type
pricePurchase
priceSale
dateOfExpiry
numberOfStocks
prescription
1
is part of
1
ProductSale
id
quantity
*
composed by
Sale
id
date
account
paid
wayToPay: wayToPay
*
made by
1
Employee
id
salary
rol:typeEmployee
*
come from
*
Provider
+ field: type
+ field: type
+ field: type
*
made by
1
Client
id
deuda
socSecNum
purchasedProducts
{Complete,
disjoint}
User
id
name
lastname1
lastname2
DNI
email
password
<<enum>>
TypeEmployee
boss
assistant
pharmacist
<<enum>>
wayToPay
creditCar
cash
<<enum>>
typeProduct
analgesics
antiallergic
anti-inflammatories
laxatives

**RN-003 Prohibido bajarle el sueldo a un empleado**

**Como** trabajador
**Quiero** que si cambiamos el sueldo de un trabajador a un salario más bajo del 20% del que tenia de error
**Para** no poder bajar los sueldos

**RN-004 No poder reservar con deudas**

**Como** trabajador
**Quiero** que un cliente que tenga una deuda superior 100 euros no pueda reservar ningún producto
**Para** que no se acumulen deudas.

**RN-005 Precio de venta distinto a precio de compra**

**Como** trabajador
**Quiero** que si se modifica el precio de algún producto, el precio de venta debe ser superior al precio de compra a los proveedores
**Para** obtener beneficios.

## 5. MODELO CONCEPTUAL

# 6. MIGRACIONES

Se encuentran en el siguiente enlace: https://github.com/mariamillan97/proyecto

1) Entidad User

```
public function up()
{
    Schema::create('users', function (Blueprint $table) {
        $table->increments('id');
        $table->string('name')->nullable();
        $table->string('lastname1')->nullable();
        $table->string('lastname2')->nullable();
        $table->string('email')->unique();
        $table->string('password');
        $table->string('DNI')->unique()->nullable;
        $table->rememberToken();
        $table->timestamps();
    });
}
```

2) Entidad Employee

```
public function up()
{
    Schema::create('employees', function (Blueprint $table) {
        $table->increments('id');
        $table->double('salary');
        $table-
>enum('typeEmployee',['boss','assistant','pharmacist']);
        $table->unsignedInteger('user_id');
        $table->foreign('user_id')->references('id')-
>on('users')->onDelete('cascade');
        $table->timestamps();
    });
}
```

3) Entidad Provider

```
public function up()
{
    Schema::create('providers', function (Blueprint $table) {
        $table->increments('id');
        $table->string('name')->nullable;
        $table->string('address');
        $table->enum('typeProduct', ['analgesics',
'antiallergic', 'anti-inflammatories', 'laxatives']);
        $table->string('email');
        $table->timestamps();
    });
}
```

4) Entidad Client

```
public function up()
{
    Schema::create('clients', function (Blueprint $table) {
        $table->increments('id');
        $table->double('deuda');
        $table->string('socSecNum')->nullable();
        $table->integer('purchasedProducts');
        $table->unsignedInteger('user_id');
        $table->foreign('user_id')->references('id')-
>on('users')->onDelete('cascade');
        $table->timestamps();
    });
}
```

5) Entidad Sale

```
public function up()
{
    Schema::create('sales', function (Blueprint $table) {
        $table->increments('id');
        $table->date('date');
        $table->double('account');
        $table->boolean('paid');
        $table->enum('waytopay',['creditCar', 'cash']);
        $table->unsignedInteger('client_id');
        $table->unsignedInteger('employee_id');
        $table->foreign('client_id')->references('id')-
>on('clients')->onDelete('cascade');
        $table->foreign('employee_id')->references('id')-
>on('employees')->onDelete('cascade');
        $table->timestamps();
    });
}
```

6) Entidad ProductSale

```
public function up()
{
    Schema::create('product_sales', function (Blueprint $table)
{
        $table->increments('id');
        $table->integer('quantity');
        $table->unsignedInteger('product_id');
        $table->unsignedInteger('sale_id');
        $table->foreign('product_id')->references('id')-
>on('products')->onDelete('cascade');
        $table->foreign('sale_id')->references('id')-
>on('sales')->onDelete('cascade');
        $table->timestamps();
    });
}
```

7) Entidad Product

```
public function up()
{
    Schema::create('products', function (Blueprint $table) {
        $table->increments('id');
        $table->string('code')->unique()->nullable();
        $table->string('name')->nullable();
        $table->enum('type',['analgesics', 'antiallergic',
'anti-inflammatories', 'laxatives']);
        $table->double('pricePurchase');
        $table->double('priceSale');
        $table->date('dateOfExpiry');
        $table->integer('numberOfStocks');
        $table->boolean('prescription');
        $table->timestamps();
    });
}
```

8) Entidad ProductProvider

```
public function up()
{
    Schema::create('product_providers', function (Blueprint
$table) {
        $table->increments('id');
        $table->unsignedInteger('product_id');
        $table->unsignedInteger('provider_id');
        $table->foreign('product_id')->references('id')-
>on('products')->onDelete('cascade');
        $table->foreign('provider_id')->references('id')-
>on('providers')->onDelete('cascade');
        $table->timestamps();
    });
}
```

## 7. RELACIONES ENTRE ENTIDADES DEL MODELO

I. App User

```
class User extends Authenticatable
{
    use Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name', 'email', 'password',
    ];

    /**
```

```
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];

    public function employee()
    {
        return $this->hasOne('App\Employee');
    }

    public function client()
    {
        return $this->hasOne('App\Client');
    }

}
```

II. App Employee

```
class Employee extends Model
{
    protected $fillable = ['user_id'];


    public function user()
    {
        return $this->belongsTo('App\User');
    }
    public function sale()
    {
        return $this->hasMany('App\Sale');
    }


}
```

III. App Provider

```
class Provider extends Model
{
    protected $fillable = ['name', 'email'];

    public function product()
    {
        return $this->belongsToMany('App\Product');
    }
}
```

IV. App Client

```
class Client extends Model
{
    protected $fillable = ['user_id', 'socSecNum'];
```

```
    public function sale()
    {
        return $this->hasMany('App\Sale');
    }

    public function user()
    {
        return $this->belongsTo('App\User');
    }
}
```

V. App Sale

```
class Sale extends Model
{
    protected $fillable = ['dale', 'client_id', 'employee_id'];

    public function productSale()
    {
        return $this->hasOne('App\ProductSale');
    }

    public function client()
    {
        return $this->belongsTo('App\Client');
    }

    public function employee()
    {
        return $this->belongsTo('App\Employee');
    }
}
```

VI. App ProductSale

```
class ProductSale extends Model
{
    protected $fillable = ['quantity', 'product_id', 'sale_id'];

    public function product()
    {
        return $this->hasOne('App\Product');
    }

    public function sale()
    {
        return $this->belongsTo('App\Sale');
    }
}
```

VII. App Product

```php
class Product extends Model

{
     protected $fillable = ['code','name','numberOfStocks'];

     public function productSale()
     {
         return $this->belongsTo('App\ProductSale');
     }

    public function provider()
    {
        return $this->belongsToMany('App\Provider');

    }

}
```

VIII. App ProductProvider

```php
class ProductProvider extends Model
{
    protected $fillable = ['product_id', 'provider_id'];
}
```