Desarrollo de un sistema web para reservas de cítas

Samuel Ricardo López Rosales

12/04/2021

1. Requerimientos del sistema

Se pretende elaborar un sistema para reservar citas dicho sistema es operado por tres roles de usarios (Administrador, Doctor y Paciente). El administrador que es el usuario más abstracto es el encargado de llenar los catalogos fuertes o catálogos principales que son el de (especialidad, medicos, pacientes) pero tambien puede aceptar o rechazar citas de todos los medicos y pacientes ademas puede ver las citas de todos los medicos y pacientes y es capas de cancelarlas si es necesario, las citas tienen tres estados principales que son (reservada, confirmada y cancelada).

Ademas el usuario administrador tiene la posibilidad de ver en un grafico la frecuencia de cítas por dia, mes etc y tambien los médicos mas activos.

Por otro lado el usuario médico tiene la posiblidad de aceptar, rechazar y cancelar citas, teiendo en cuenta que al cancelar una cita antes de ser confirmada no se pide una explicación, pero al cancelar despues de la cita ser confirmada se tiene que dar una explicación, tambien tiene la posiblidad de mirar sus citas y puede tambien gestionar su horario activando dias e ingresando la hora de inicio y fin del turno mañana y tambien el inicio y fin del turno de la tarde.

Mientras que el usuario paciente tiene la posiblidad de solicitar una cita ingresando la especialidad del médico, seleccionando el medico y seleccionando la fecha y hora disponible. Tambien puede ver sus citas y puede cancelar si es necesario.

El usuario paciente tambien puede observar sus citas pero no puede hacer modificaciones y puede observar sus citas y status.

2. Base de datos

3. Creación del proyecto Laravel

Para crear el proyecto con una versión expecifica existen diferentes comandos y uno es este composer create-project laravel/laravel nombre "5.3"

3.1. Migraciones y autenticación

En el archivo .env es necesario especificar las credenciales de la base de datos que vamos a utilizar, para crear una autenticacion y registroa utomaticos se ingresa el siguiente comando "php artisan make:auth". Automaticamente nos crea la migración user y tambien la tabla password-reset y tambien nos crea un LoginController y un RegisterController. Para ejecutar las migraciones presionamos "php artisan migrate".

3.2. Integración de la plantilla Argon

Como ya tenemos un login y un registro trabajaremos con ellos solo que les incluiremos un diseño diferenete deacuerdo a la plantilla Argon. utilizamos la directiva @yield en el padre para que tenga una sección modificable, y en la vista hija ponemos @section y @endsection, si ponemos @yield('parametro1','parametro2') estamos diciendo que si no se incluye en el hijo el section entonces ponga por defecto el segundo parametro. Tambien existe la directiva @include('ruta del archivo') es para agregar fragmentos de elementos. Tambien para acceder a un elemento del usuario que ingreso se utiliza {{ auth()→user()→name }}

3.3. Crear repositorio de Git

(git init) es para iniciar el repositorio y (git add .) sirve para agregar los cambios y finalmente creamos un commit (git commit -m "Base project").

4. Gestor de datos (Specialty y User)

Para crear un modelo, una migración y un controlador al mismo tiempo "php artisan make:model Specialty -mc"

```
1  public function up()
2  {
3    Schema::create('specialties', function (Blueprint $table) {
4      $table->id();
5      $table->string('name');
6      $table->string('description')->nullable();
7      $table->timestamps();
8    });
9  }
10
```

Listing 1: Código de la migración de Specialty

Despues se crean las rutas que se utilizaran para crear el CRUD de especialidades.

```
1    Route::get('/specialties','SpecialtyController@index');
2    Route::get('/specialties/create','SpecialtyController@create');
3    Route::get('/specialties/{specialty}/edit','
     SpecialtyController@edit');
4    Route::post('/specialties', 'SpecialtyController@store');
5    Route::put('/specialties/{specialty}','
     SpecialtyController@update');
6    Route::delete('/specialties/{specialty}','
     SpecialtyController@destroy');
7
```

Listing 2: Rutas de Specialty

Cuando queremos llamar una ruta si ya tiene un nombre asignado se pone {{ route('home') }}, pero si no tiene nombre se manda llamar {{ url('specialties/create') }}

Para crear una especialidad es necesario poner la etiqueta formulario con el action hacia a donde va apuntar y tambien los input con el name especifico y el boton de submit.

```
<form action="{{ url('specialties') }}" method="POST">
@csrf
<div class="form-group">
<label for="name">Nombre de la especialidad</label>
<input type="text" name="name" class="form-control" value="{{
old('name') }}" required>
</div>
<div class="form-group">
<label for="name">Descripcion</label>
<input type="text" name="description" class="form-control">
</div>

<button type="submit" class="btn btn-primary">Guardar</button>
</form>

```

Listing 3: Formulario Especialidad

Como se obsera esté formulario manda llamar el metodo Store() y recibe un request que son los nombres de los input.

```
public function store(Request $request){
  //dd($resquest->all());
  $this->performValidation($request);
  // si hay un error lo retorna y bloquea el flujo

  $specialty = new Specialty();
  $specialty->name = $request->input('name');
  $specialty->description = $request->input('description');
  $specialty->save();  // realiza un insert en la tabla
specialties

  $notificacion = 'La especialidad se ha registrado
correctamente';
  return redirect('/specialties')->with(compact('notificacion')
);// la variable se va a crear como una variable de sesion
}

```

Listing 4: Metodo para crear especialidad

Para listar todas las especialidades en la tabla se manda a llamar el metodo index.

```
public function index(){
  // obtener el listado de espacialidades
  $specialties = Specialty::all();
  return view('specialties.index',compact('specialties'));
}

```

Listing 5: obtener toda la lista de especialidades

para pintar los datos en la tabla es necesario crear un foreach para recorrer el arraeglo.

```
 @foreach($specialties as $specialty)
<tr>
<th scope="row">
{{ $specialty->name }}
</th>
<td>
{{ $specialty->description }}
</td>
```

```
    <td>

    <form action="{{ url('specialties/'.$specialty->id) }}" method
    ="POST">
    @csrf
    @method('DELETE')
    <a href="{{ url('specialties/'.$specialty->id.'/edit') }}"
    class="btn btn-sm btn-primary">Editar</a>
    <button type="submit" class="btn btn-sm btn-danger">Eliminar</
    button>
    </form>
    </td>
    </tr>
    @endforeach

```

Listing 6: pintar datos en tabla

Para realizar una validación es necesario utilizar el objeto validate que ya proporciona Laravel, a continuación se muestra que se requieren tres parametros, el primer parametro son el arreglo de datos, el segundo son las reglas a aplicar y finalmente el tercero son los mensajes que se muestran en caso de que se active un error, tambien se muestra como mostrar los mensajes ya que nos devuelve una variable de sesion.

```
    $rules = [
    'name' => 'required|min:3'
    ];
    $messages = [
    'name.required' => 'Es necesario ingresar el nombre ',
    'name.min' => 'Como minimo el nombre debe tener 3 caracteres'
    ];
    $this->validate($request,$rules, $messages);

    /****************** MOSTRAR LOS ERRORES *****************/
    <div class="card-body">
    @if ($errors->any())
    <div class="alert alert-danger" role="alert">
    <ul>
    @foreach($errors->all() as $error)
    <li>{{ $error }}</li>
    @endforeach
    </ul>
    </div>
    @endif

```

Listing 7: Validaciones del lado servidor

4.1. Actualizar datos de la especialidad

Primero mandamos llamar la ruta del método edit para que nos devuleva la especialidad seleccionada y nos permita pintar los datos en la vsta, se observ que recibe un parametro de tipo Specialty, si le enviamos el Id el mismo framework se encargara de castear para que sea un objeto Specialty, pero si le decimos que reciba otro tipo de parametro, solo recibiremos un numero.

```
    public function edit(Specialty $specialty){
      return view('specialties.edit', compact('specialty'));
    }
```

```
4
5
6     /************************* MOSTRAR LOS DATOS DE LA
      ESPECIALIDAD *****************/
7     <form action="{{ url('specialties/'.$specialty->id ) }}" method
      ="POST">
8     @csrf
9     @method('PUT')
10    <div class="form-group">
11    <label for="name">Nombre de la especialidad</label>
12    <input type="text" name="name" class="form-control" value="{{
      old('name',$specialty->name ) }}" required>
13    </div>
14    <div class="form-group">
15    <label for="name">Descripcion</label>
16    <input type="text" name="description" class="form-control"
      value="{{ old('description',$specialty->description ) }}">
17    </div>
18
19    <button type="submit" class="btn btn-primary">Guardar</button>
20    </form>
21
```

Listing 8: funcion edit

Despues al presionar el boton de guardar cambios se manda llamar la ruta update que tambien recibe un request en el siguiente código se muestra esté método.

```
1      public function update(Request $request, Specialty $specialty)
      {
2       //dd($resquest->all());
3       $this->performValidation($request);
4
5       $specialty->name = $request->input('name');
6       $specialty->description = $request->input('description');
7       $specialty->save();  // realiza un UPDATE en la tabla
      specialties
8       $notificacion = 'La especialidad se ha actualizado
      correctamente';
9       return redirect('/specialties')->with(compact('notificacion')
      );
10    }
11
```

Listing 9: funcion update

Finalmente realizaremos la funcionalidad de eliminar especialidades, utilizando el metodo DELETE se creo un boton que esta dentro de un formulario.

```
1     <form action="{{ url('specialties/'.$specialty->id) }}" method
      ="POST">
2     @csrf
3     @method('DELETE')
4     <a href="{{ url('specialties/'.$specialty->id.'/edit') }}"
      class="btn btn-sm btn-primary">Editar</a>
5     <button type="submit" class="btn btn-sm btn-danger">Eliminar</
      button>
6     </form>
7
8     /********************** FUNCION EN EL CONTROLADOR
      **********************/
9      public function destroy(Specialty $specialty){
```

```
10      $deleteName = $specialty->name;
11      $specialty->delete(); # eliminar una especialidad
12      $notificacion = 'La especialidad '.$deleteName.' se ha
    eliminado correctamente';
13      return redirect('/specialties')->with(compact('notificacion')
    );
14    }
15
```

Listing 10: funcion update

5. Gestionar datos de usuario (Doctor, Patient, Administrador)

Laravel nos da un modelo usuario por defecto pero es necesario agregar más campos como la dirección, telefono, dni, y el rol, vamos a usar el mismo modelo User y vamos hacer dos scopes para filtrar doctores y pacientes. Primero que nada vamos a crear las rutas cabe mencionar que con un Route::resource(") se crean todos los datos CRUD.

```
1   // DOCTORS
2   Route::resource('doctors','DoctorController');
3   Route::resource('patients','PatientController');
4
```

Listing 11: Ruta doctor y patients

Despues vamos a crear los controladores, en este caso crearemos un controlador con todos los metodos ya incluidos, para esto se teclean los siguientes comandos "php artisan make:controller DoctorController –resources".

5.1. Crear Seeder y factories y faker para ususarios

Laravel nos crea un DataBaseSeeder en la carpeta seeds/DataBaseSeeder y desde aquí se mandan a llamar todos los seeders, para crear un seed se teclea lo siguiente "php artisan make:seeder UsersTableSeeder", tambien laravel ya nos crea por defecto un UserFactory, que es el modelo. Dentro esta factory/UserFactry, nos va a dar los datos de dorma aleatoria.

```
1   $factory->define(User::class, function (Faker $faker) {
2     return [
3     'name' => $faker->name,
4     'email' => $faker->unique()->safeEmail,
5     'email_verified_at' => now(),
6     'password' => '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og
    /at2.uheWG/igi', // password
7     'remember_token' => Str::random(10),
8     'dni' => $faker->randomNumber(8,true),
9     'address' => $faker->address,
10    'phone' => $faker->e164PhoneNumber,
11    'role' => $faker->randomElement(['doctor','patient'])
12    ];
13  });
14
15  // DDEFINIR ESTADO
16  $factory->state(App\User::class, 'patient', [
17  'role' => 'patient'
```

```
18      ]);
19
20      $factory->state(App\User::class, 'doctor', [
21      'role' => 'doctor'
22      ]);
23
```

Listing 12: ModelFactory

Despues de crear el modelFactory es necesario llamar ese model desde el seeder para crear usuarios random, dicho archivo esta en seeds/UserTableSeeder.

```
1   public function run()
2   {
3     User::create([
4     'name' => 'admin',
5     'email' => 'samuellopezrosales101@gmail.com',
6     'password' => bcrypt('Barcelona#123'),
7     'dni' => '12345678',
8     'address' => '',
9     'phone' => '',
10    'role' => 'admin'
11    ]);
12    User::create([
13    'name' => 'medico',
14    'email' => 'samy-messi10@hotmail.com',
15    'password' => bcrypt('Barcelona#123'),
16    'dni' => '12345678',
17    'address' => '',
18    'phone' => '',
19    'role' => 'doctor'
20    ]);
21    User::create([
22    'name' => 'Paciente',
23    'email' => 'paciente@paciente.com',
24    'password' => bcrypt('Barcelona#123'),
25    'dni' => '12345678',
26    'address' => '',
27    'phone' => '',
28    'role' => 'patient'
29    ]);
30    factory(User::class,50)->states('patient')->create();
31  }
32
```

Listing 13: SeederUsuario

Para que se ejecute finalmente se manda llamar el seed desde DatabaseSeeder seeds/DataBaseSeeder. finalmente se teclea el siguiete comando "php artisan migrate:refresh –seed", esto va hacer rollback de las migraciones es decir las tablas van a ser eliminadas y se volveran a crear y finalmente se ejecutaran los seeders.

```
1   class DatabaseSeeder extends Seeder
2   {
3     /**
4     * Seed the application's database.
5     *
6     * @return void
7     */
8     public function run()
9     {
10      $this->call([
```

```
11      UserTableSeeder::class,
12      SpecialtiesTableSeeder::class,
13      WorkDaysTableSeeder::class
14      ]);
15
16    }
17  }
18
```

Listing 14: Ejecutar Seed

Para filtrar los usuarios por rol se manda llamar User::where('role','patient')→get(), pero tambien se puede hacer un Scope para hacer el filtro y no estar llamando al where(), un Scope se realiza dentro de un modelo en este caso dentro del modelo User.

```
1       public function scopePatients($query){
2         return $query->where('role','patient');
3       }
4
5       public function scopeDoctors($query){
6         return $query->where('role','doctor');
7       }
8
9     /************************ MANDARLO LLAMAR EN EL CONTROLADOR
      ****************************/
10
11     public function index()
12    {
13      $doctors = User::doctors()->get();
14      return view('doctors.index',compact('doctors'));
15    }
16
17  public function index()
18  {
19    $patients = User::patients()->paginate(10);
20    return view('patients.index',compact('patients'));
21  }
22
```

Listing 15: Scope doctors y patients

Hay una forma de proteger los datos que ingresamos con un only() solo se tomaran encuenta los campos que proporcionamos, pero tambien hay que asignar en el modelo que campos se pueden usar de manera masiva.

```
1       /************************ ESTO EN EL MODELO
    ***************************/
2
3       // CAMPOS QUE SE ASIGNAN DE MANERA MASIVA
4       protected $fillable = [
5       'name', 'email', 'password','dni','address','phone','role'
6       ];
7       // CAMPOS QUE ESTAN OCULTOS Y NO SE PUEDEN CONSULTAR //
8       protected $hidden = [
9       'password', 'remember_token','pivot'
10      ];
11
12      /************************CONTROLADOR CREAR MEDICO
    *****************/
13      public function store(Request $request)
14      {
15        $rules = [
```

```
        'name' => 'required|min:3',
        'email' => 'required|email',
        'dni' => 'nullable|digits:8',
        'address' => 'nullable|min:5',
        'phone' => 'nullable|min:6'
        ];
        $this->validate($request,$rules);

        $user = User::create(
        $request->only('name','email','dni','address','phone')
        + [
        'role' => 'doctor',
        'password' => bcrypt($request->input('password'))
        ]
        );
      }



      $notificacion = 'El Medico se ha registrado correctamnete.';
      return redirect('/doctors')->with(compact('notificacion'));

```

Listing 16: Crear doctor

Tambien se puede mandar a llamar un medico buscandolo en el controlador solo se pasara su id y con la funcion find() se busca.

```
    public function edit($id)
    {
      $doctor = User::doctors()->findOrFail($id);
      return view('doctors.edit',compact('doctor'));
    }

```

Listing 17: Buscar doctor

La contraseña no simpre será modificada por eso si existe un valor de l campo password se agregara al data y si no se quedara vacio, con el metodo fill vamos agregar nueva información.

```
     public function update(Request $request, $id)
    {
      $rules = [
      'name' => 'required|min:3',
      'email' => 'required|email',
      'dni' => 'nullable|digits:8',
      'address' => 'nullable|min:5',
      'phone' => 'nullable|min:6'
      ];
      $this->validate($request,$rules);

      $user = User::doctors()->findOrFail($id); // busco el objeto
      $data = $request->only('name','email','dni','address','phone
    ');
      $password = $request->input('password');
      if($password)
      $data['password'] = bcrypt($password);

      $user->fill($data);
      $user->save();

      $user->specialties()->sync($request->input('specialties'));
```

```
22
23      $notificacion = 'El Medico se ha actualizado correctamnete.';
24      return redirect('/doctors')->with(compact('notificacion'));
25    }
26
27
```

Listing 18: Actualizar Medico

Que diferencia hay entre un médico y un paciente el médico esta asociado a especialidades, y son los que gestionan el horario, y el paciente es quien ve los horarios de los médicos.

6. Proteger rutas usando Middlewares

Laravel ya define por nosotros el middleware de autenticación, para nosotros crear un middleware tecleamos el siguiente comando "php artisan make:middleware AdminMiddleware", si el usuario tiene rol admin le permitiremos continuar, pero si no vamos a redirigir al usuario hacia la ruta de inicio. En este caso haremos una agrupación de rutas y aplicar distintos middlewares.

```
1     public function handle($request, Closure $next)
2     {
3       if(auth()->user()->role == 'admin')
4       return $next($request);
5       return redirect('/');
6     }
7
8
9     /******************* EJECUTAR MIDDLEWARE GRUPO DE RUTAS
      ***************/
10    Route::middleware(['auth','admin'])->namespace('Admin')->group(
      function(){
11      Route::get('/specialties','SpecialtyController@index');
12      Route::get('/specialties/create','SpecialtyController@create
      ');
13      Route::get('/specialties/{specialty}/edit','
      SpecialtyController@edit');
14
15      Route::post('/specialties', 'SpecialtyController@store');
16      Route::put('/specialties/{specialty}','
      SpecialtyController@update');
17      Route::delete('/specialties/{specialty}','
      SpecialtyController@destroy');
18
19
20      // DOCTROS
21      Route::resource('doctors','DoctorController');
22      Route::resource('patients','PatientController');
23
24      Route::get('/charts/appointments/line','
      ChartController@appointments');
25      Route::get('/charts/doctors/bar','ChartController@doctors');
26    });
27
28
```

Listing 19: Middleware Rol Admin

Pero antes tenemos que registrar el middleware app/http/kernel.php.

```
    protected $routeMiddleware = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'auth.basic' => \Illuminate\Auth\Middleware\
    AuthenticateWithBasicAuth::class,
    'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings
    ::class,
    'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders
    ::class,
    'can' => \Illuminate\Auth\Middleware\Authorize::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'password.confirm' => \Illuminate\Auth\Middleware\
    RequirePassword::class,
    'signed' => \Illuminate\Routing\Middleware\ValidateSignature::
    class,
    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::
    class,
    'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified
    ::class,
    'admin' => \App\Http\Middleware\AdminMiddleware::class,
    'doctor' => \App\Http\Middleware\DoctorMiddleware::class,
    ];

```

Listing 20: Registrar middleware en kernel

7. Gestionar Horarios de los médicos

Primero que nada debemos crear un controlador para el horario llamado ScheduleController, "php artisan make:controller Doctor ScheduleController ", y tambien vamos a definir las rutas para este Controlador.

```
  /************* DEFINIR RUTAS *********/

  Route::middleware(['auth','doctor'])->namespace('Doctor')->group(
    function(){
    Route::get('/schedule','ScheduleController@edit');
    Route::post('/schedule','ScheduleController@store');
  });


```

Listing 21: ScheduleController edit

Tambien vamos a crear una tabla llamada workDays que contiene los campos (id, day(int 0-6), active(boolean), morning-start, morning-end, afternoon-start, afternoon-end, user-id).

```
    public function up()
    {
      Schema::create('work_days', function (Blueprint $table) {
        $table->id();
        $table->unsignedSmallInteger('day');
        $table->boolean('active');

        $table->time('morning_start');
        $table->time('morning_end');

        $table->time('afternoon_start');
        $table->time('afternoon_end');

```

```
        $table->unsignedBigInteger('user_id');
        $table->foreign('user_id')->references('id')->on('users');

        $table->timestamps();
      });
    }

```

Listing 22: ScheduleController edit

Primero creo un arreglo de los 7 dias ['lunes','martes',.....], pero tambien vamos a buscar los dias de trabajo deacuerdo a ese usuario, mediante el user-id, en caso de tener datos, esos datos los vamos a convertir a objetos Carbon con un formato ya determinado, con la función map se hace eso se cambian las que se tienen y se agregan las nuevas ya formateadas y en caso de no tener horario de trabajo se crean 7 dias. Tambien cabe señalar que utilizamos el metodo in-array que busca un elemento dentro de un array.

```
    private $days = ['Lunes','Martes','Miercoles','Jueves','Viernes
    ','Sabado','Domingo'];
    public function edit(){

      $workDays = WorkDay::where('user_id',auth()->user()->id)->get
    ();
      // la siguiente funcion me permite recibir arreglo y
    convertirlo a otro
      if(count($workDays) > 0){
        $workDays->map(function($workDay){
          $workDay->morning_start = (new Carbon($workDay->
    morning_start))->format('g:i A');
          $workDay->morning_end = (new Carbon($workDay->morning_end
    ))->format('g:i A');
          $workDay->afternoon_start = (new Carbon($workDay->
    afternoon_start))->format('g:i A');
          $workDay->afternoon_end = (new Carbon($workDay->
    afternoon_end))->format('g:i A');
          return $workDay;
        });
      }else{
        $workDays = collect();
        for($i=0; $i<7; $i++)
        $workDays->push(new WorkDay());
      }
      $days = $this->days;
      return view('schedule',compact('workDays','days'));
    }



    /****************** EN LA VISTA SE MUESTRAN **************/

    <table class="table align-items-center table-flush">
    <thead class="thead-light">
    <tr>
    <th scope="col">Dias</th>
    <th scope="col">Activo</th>
    <th scope="col">Turno Manana</th>
    <th scope="col">Turno Tarde</th>
    </tr>
    </thead>
    <tbody>
    @foreach($workDays as $key => $workDay)
```

```
38      <tr>
39      <td>{{ $days[$key] }}</td>
40      <td>
41      <label class="custom-toggle">
42      <input type="checkbox" name="active[]" value="{{ $key }}"
43      @if($workDay->active) checked @endif>
44      <span class="custom-toggle-slider rounded-circle"></span>
45      </label>
46      </td>
47      <td>
48      <div class="row">
49      <div class="col">
50      <select class="form-control" name="morning_start[]">
51      @for($i=5; $i<=11; $i++)
52      <option value="{{ ($i<10 ? '0' : '') . $i }}:00" @if($i.':00 AM
        ' == $workDay->morning_start) selected @endif>{{ $i }}:00 AM</
        option>
53      <option value="{{ ($i<10 ? '0' : '') . $i }}:30" @if($i.':30 AM
        ' == $workDay->morning_start) selected @endif>{{ $i }}:30 AM</
        option>
54      @endfor
55      </select>
56      </div>
57      <div class="col">
58      <select class="form-control" name="morning_end[]">
59      @for($i=5; $i<=11; $i++)
60      <option value="{{ ($i<10 ? '0' : '') . $i }}:00" @if($i.':00 AM
        ' == $workDay->morning_end) selected @endif>{{ $i }}:00 AM</
        option>
61      <option value="{{ ($i<10 ? '0' : '') . $i }}:30" @if($i.':30 AM
        ' == $workDay->morning_end) selected @endif>{{ $i }}:30 AM</
        option>
62      @endfor
63      </select>
64      </div>
65      </div>
66      </td>
67      <td>
68      <div class="row">
69      <div class="col">
70      <select class="form-control" name="afternoon_start[]">
71      @for($i=1; $i<11; $i++)
72      <option value="{{ $i+12 }}:00" @if($i.':00 PM' == $workDay->
        afternoon_start) selected @endif>{{ $i }}:00 pm</option>
73      <option value="{{ $i+12 }}:30" @if($i.':30 PM' == $workDay->
        afternoon_start) selected @endif>{{ $i }}:30 pm</option>
74      @endfor
75      </select>
76      </div>
77      <div class="col">
78      <select class="form-control" name="afternoon_end[]">
79      @for($i=1; $i<11; $i++)
80      <option value="{{ $i+12 }}:00" @if($i.':00 PM' == $workDay->
        afternoon_end) selected @endif>{{ $i }}:00 pm</option>
81      <option value="{{ $i+12 }}:30" @if($i.':30 PM' == $workDay->
        afternoon_end) selected @endif>{{ $i }}:30 pm</option>
82      @endfor
83      </select>
84      </div>
85      </div>
86      </td>
87      </tr>
```

```
88      @endforeach
89      </tbody>
90      </table>
91
```

Listing 23: ScheduleController edit

Toda la tabla se metio dentro de un form que apunta a la ruta post('schedule'), tambien se observa que el name=morning-start[] son arreglos y llegarian a al metodo store puros arreglos. , tambien hay que verificar si es la primera vez que el médico va ingresar su horario tenemos que hacer una operación insert, pero si ya tiene un horario hay que actualizar, para eso utilizamos el método updateOrCreate(), esté método recibe dos arreglos el primero son los datos a buscar, si no estan esos datos creo un regsitro con los datos del segundo arreglo y si si estan solo actaliza con los datos del segundo arreglo. Tambien se realiza una validación de la consistencia de las horas vemos si las horas de inicio son mayores a las horas de fin entonces se manda una alerta.

```
1       public function store(Request $request){
2        //dd($request->all());
3        //die();
4        $active = $request->input('active') ?: []; // si no vienen
    elementos  asigna un array vacio
5        $morning_start = $request->input('morning_start');
6        $morning_end = $request->input('morning_end');
7        $afternoon_start = $request->input('afternoon_start');
8        $afternoon_end = $request->input('afternoon_end');
9
10       $errors = [];
11       for($i=0; $i<7; $i++){
12         if($morning_start[$i] > $morning_end[$i]){
13           $errors [] = "Las horas del turno de la manana son
    incosistentes para el dia " . $this->days[$i] . " .";
14         }
15
16         if($afternoon_start[$i] > $afternoon_end[$i]){
17           $errors [] = "Las horas del turno de la tarde son
    incosistentes para el dia ". $this->days[$i]. " .";
18         }
19         WorkDay::updateOrCreate(
20         ['day' => $i, // son elementos que hay que buscar para ver
    si ya estan
21         'user_id' => auth()->id()],
22         [
23         'active' => in_array($i, $active), // si esta el elemento
    retorna 1 y si no un cero
24         'morning_start' => $morning_start[$i],
25         'morning_end' => $morning_end[$i],
26         'afternoon_start' => $afternoon_start[$i],
27         'afternoon_end' => $afternoon_end[$i]
28
29         ]
30         );
31       }
32
33       if(count($errors) > 0)
34       return back()->with(compact('errors'));
35
36       $notificacion = 'Los cambios se han guardado correctamente.';
37       return back()->with(compact('notificacion'));
38
```

```
39      }
40
```

Listing 24: ScheduleController store(guardar horario)

8. Registrar cita versión web(paciente)

Para registrar una cita fue necesesario crear una tabla para citas llamadas appointments, se teclea el siguiente comando para crear una migracion, controlador y modelo "php artisan make:model Appointment -mc". pero tambien tenemos que registrar una ruta para mandarlos llamar.

```
1     /******** RUTAS *************/
2     Route::get('/appointments/create','AppointmentController@create
      ');
3     Route::post('/appointments','AppointmentController@store');
4
5     /******* MIGRACION **************/
6     public function up()
7     {
8       Schema::create('appointments', function (Blueprint $table) {
9         $table->id();
10
11        $table->string('description');
12
13        // fk_ specialty
14        $table->unsignedBigInteger('specialty_id');
15        $table->foreign('specialty_id')->references('id')->on('
      specialties');
16
17        // fk_ doctor
18        $table->unsignedBigInteger('doctor_id');
19        $table->foreign('doctor_id')->references('id')->on('users')
      ;
20
21        // fk_ specialty
22        $table->unsignedBigInteger('patient_id');
23        $table->foreign('patient_id')->references('id')->on('users
      ');
24
25        $table->date('scheduled_date');
26        $table->time('scheduled_time');
27
28        $table->string('type');
29
30
31
32        $table->timestamps();
33      });
34    }
35
36
37    /****** el controlador retorna todas las especialidades *****/
38    public function create(){
39      $specialties = Specialty::all();
40      return view('appointments.create', compact('specialties'));
41    }
42
43    /************ pinta las especialidades en un select ****/
44    <div class="form-group col-md-6">
```

```
45      <label for="name">Especialidad</label>
46      <select name="specialty_id" id="specialty" class="form-control"
         required>
47      <option value="">Seleccione especialidad</option>
48      @foreach($specialties as $key => $specialty)
49      <option value="{{ $specialty->id }}" @if(old('specialty_id')==
        $specialty->id)
50      selected @endif>{{ $specialty->name }}</option>
51      @endforeach
52      </select>
53      </div>
54
```

Listing 25: ruta,migracion, migracion y controlador cita

8.1. Relación muchos a muchos entre médico y especialidad

Un medico puede tener muchas especialidades y una especialidad puede tener muchos médicos por eso es una relación muchos a muchos, ahora es necesario decirselo a laravel mediante el ORM, por dicho motivo se hara una tabla intermedia que contenga el id de la esepcialidad y el id del médico, podemos elegir muchas especialidades con bootstrap select. (Laravel sabe con que tabla estan cada modelo lo que hace es usar el nombre del modelo y le agrega el plural), si no tuviera el mismo nombre tendriamos que indicarle a laravel el nombre de a tabla que queremos usar con protected table = 'nombre de tabla', tambien hay una convención para las lleves foraneas y para tablas intermedias, para definir un atabla intermedia ambos modelos deben tener el nombre de la tabla en singular y se forman de forma alfabetica (specialty-user), va a buscar la tabla y le agrgara un id user-id, specialty-id, para crear esto teclearemos "php artisan make:migration create-specilaty-user-table". Para definir la relación hay que definir 2 metodos uno en el modelo user que se llame specialties y otro en specialty que se llame users. si no hubieramos utilizado la convención hubiermaos tenido que definir en el metodo tambine el nombre de la tabla intermedia y el nombre del identificador.

```
1       public function up()
2       {
3         Schema::create('specialty_user', function (Blueprint $table)
        {
4           $table->id();
5           // doctor
6           $table->unsignedBigInteger('user_id');
7           $table->foreign('user_id')->references('id')->on('users');
8           // especialidad
9           $table->unsignedBigInteger('specialty_id');
10          $table->foreign('specialty_id')->references('id')->on('
        specialties');
11
12          $table->timestamps();
13        });
14      }
15
16      /******** MODELO USER ***************/
17
18      public function specialties()
19      {
```

```
20      return $this->belongsToMany(Specialty::class)->withTimestamps
    ();
21    }
22
23    /******** MODELO SPECIALTY *********/
24    public function users()
25    {
26      return $this->belongsToMany(User::class)->withTimestamps();
27    }
28
```

Listing 26: relacion specialty y user

8.2. Modificación para crear un médico para asignarle las especialidades que requiere

Para seleccionar especilaidades para un médico necesitamos usar el bootstrap-select, que e sun plugin externo, y entoces lo que recibira el strore para crear el médico sera un arreglo de especialidades. Desde el modelo de usuarios tenemos specialties y tambien tenemos el metodo attach que se encarga de crear relaciones y ya solo es necesario pasarle el arreglo de especialidades.

```
1     <div class="form-group">
2     <label for="specialties">Especialidades</label>
3     <select name="specialties[]" id="specialties" class="form-
      control selectpicker" data-style="btn-default" multiple title="
      Seleccione una o varias">
4     @foreach($specialties as $specialty)
5     <option value="{{ $specialty->id }}">{{ $specialty->name }}</
      option>
6     @endforeach
7     </select>
8     </div>
9
10
11    /********* METODO PARA CREAR MEDICO****/
12    public function store(Request $request)
13    {
14      $rules = [
15      'name' => 'required|min:3',
16      'email' => 'required|email',
17      'dni' => 'nullable|digits:8',
18      'address' => 'nullable|min:5',
19      'phone' => 'nullable|min:6'
20      ];
21      $this->validate($request,$rules);
22
23      $user = User::create(
24      $request->only('name','email','dni','address','phone')
25      + [
26      'role' => 'doctor',
27      'password' => bcrypt($request->input('password'))
28      ]
29      );
30      // asignar especialidades al medico
31      $user->specialties()->attach($request->input('specialties'));
       // va a recibir el arreglo de las especilaidadex
32
33      $notificacion = 'El MEdico se ha registrado correctamnete.';
34      return redirect('/doctors')->with(compact('notificacion'));
```

```
35      }
36
```

Listing 27: Asignar especialidades a un medico

Para listar los médicos segun la especialidad tenemos varias opciones tales como usar (vue, jquery, javascript), en esté caso utilizaremos jQuery. cuando laravel devuelve una coleccoón la devuelve automaticamente en JSON.

```
1       Route::get('/specialties/{specialty}/doctors','Api\
        SpeicaltyController@doctors');
2
3       /**** retornar los doctores de esta especialidad ******/
4
5       public function doctors(Specialty $specialty)
6       {
7         return $specialty->users()->get([
8         'users.id','users.name'
9         ]);
10      }
11
```

Listing 28: Ruta medicos segun la especialidad

Cada que exista un cambio en el campo specialty mandaremos llamar la ruta para listar los médicos segun la especialidad para eso utilizamos el método change() y llenamos el otro select con los datos que me trae la ruta.

```
1       /*********** HTML MEDICOS ************/
2
3       <div class="form-row">
4       <div class="form-group col-md-6">
5       <label for="name">Especialidad</label>
6       <select name="specialty_id" id="specialty" class="form-control"
         required>
7       <option value="">Seleccione especialidad</option>
8       @foreach($specialties as $key => $specialty)
9       <option value="{{ $specialty->id }}" @if(old('specialty_id')==
        $specialty->id)
10      selected @endif>{{ $specialty->name }}</option>
11      @endforeach
12      </select>
13      </div>
14      <div class="form-group col-md-6">
15      <label for="email">Medico</label>
16      <select name="doctor_id" id="doctor" class="form-control"
        required>
17      @foreach($doctors as $doctor)
18      <option value="{{ $doctor->id }}" @if(old('doctor_id')==
        $doctor->id)
19      selected @endif>{{ $doctor->name }}</option>
20      @endforeach
21      </select>
22      </div>
23      </div>
24
25      /************ JAVASCRIPT PARA LLENAR SELECT ********/
26      $(function() {
27        $specialty = $('#specialty');
28        $doctor = $('#doctor');
29        $date = $('#date');
30        $hours = $('#hours');
31
```

```
32      $specialty.change(() => {
33        const specialtyId = $specialty.val();
34        const url = `/specialties/${specialtyId}/doctors`;
35        $.getJSON(url,onDoctorsLoaded);
36      });
37
38      $doctor.change(loadHours);
39      $date.change(loadHours);
40    });
41
42
43    function onDoctorsLoaded(doctors){
44      let htmlOptions = '';
45      doctors.forEach(doctor => {
46        htmlOptions += `<option value="${doctor.id}">${doctor.name
    }</option>`;
47      });
48      $doctor.html(htmlOptions);
49      loadHours();
50    }
51
```

Listing 29: Llenar medicos segun la especialidad

8.3. Obtener horas por intervalo de 30 minutos segun medico y fecha

Para eso craremos un controlador llamado ScheduleController y vamos a definir una ruta que nos retorne la shoras.

```
1     /*********** RUTA ************/
2
3     Route::get('/schedule/hours','Api\ScheduleController@hours');
4
5     /**************** METODO HOURS **************/
6
7     public function hours(Request $request,
      ScheduleServiceInterface $scheduleService){
8       $rules =[
9       'date'=>'required|date_format:"Y-m-d"',
10      'doctor_id'=>'required|exists:users,id'
11      ];
12      $this->validate($request, $rules);
13
14      $date = $request->input('date');
15      $doctorId = $request->input('doctor_id');
16
17
18      return $scheduleService->getAvailableIntervals($date,
      $doctorId);
19
20    }
21
22
23    /*********** INTERFACE **************************/
24    interface ScheduleServiceInterface
25    {
26      public function isAvailableInterval($date, $doctorId, Carbon
      $start);
27      public function getAvailableIntervals($date, $doctorId);
28    }
```

```
29
30
31    /********** SERVICE ****************************/
32
33    class ScheduleService implements ScheduleServiceInterface
34    {
35      public function isAvailableInterval($date, $doctorId, Carbon
      $start){
36          $exists = Appointment::where('doctor_id', $doctorId)
37          ->where('scheduled_date', $date)
38          ->where('scheduled_time', $start->format('H:i:s'))
39          ->exists();
40          return !$exists;
41        }
42        public function getDayFromDate($date)
43        {
44          $dateCarbon= new Carbon($date);
45
46          $i = $dateCarbon->dayOfWeek;
47          $day = ($i==0 ? 6 :$i-1);
48          return $day;
49        }
50        public function getAvailableIntervals($date, $doctorId)
51        {
52          $workDay = WorkDay::where('active', true)
53          ->where('day', $this->getDayFromDate($date))
54          ->where('user_id', $doctorId)
55          ->first([
56          'morning_start', 'morning_end',
57          'afternoon_start', 'afternoon_end'
58          ]);
59          if(!$workDay){
60            return [];
61          }
62          $morningIntervals = $this->getIntervals(
63          $workDay->morning_start, $workDay->morning_end, $date,
      $doctorId);
64          $afternoonIntervals = $this->getIntervals(
65          $workDay->afternoon_start, $workDay->afternoon_end
66          ,$date, $doctorId
67          );
68          $data = [];
69          $data['morning']= $morningIntervals;
70          $data['afternoon']= $afternoonIntervals;
71          return $data;
72        }
73
74        private function getIntervals($start, $end, $date, $doctorId)
      {
75          $start = new Carbon($start);
76          $end = new Carbon($end);
77
78          $intervals = [];
79          while($start < $end){
80            $interval = [];
81
82            $interval['start'] = $start->format('g:i A');
83
84            // no existe una cita para esta hora con este medico
85            $available = $this->isAvailableInterval($date, $doctorId,
       $start);
86
```

```
87          $start->addMinutes(30);
88          $interval['end'] = $start->format('g:i A');
89
90          if($available){
91            $intervals [] = $interval;
92          }
93        }
94        return $intervals;
95      }
96    }
97
```

Listing 30: Rangos de Horas JSON

La ruta previamente propuesta la vamos a mandar llamar cada que exista un cambio ya sea en el campo medico o en el campo schedule-date y entonces pintaremos los intervalos.

```
1   function loadHours(){
2     const selectedDate = $date.val();
3     const doctorId = $doctor.val();
4     const url = `/schedule/hours?date=${selectedDate}&doctor_id=$
    {doctorId}`;
5     $.getJSON(url, displayHours);
6   }
7
8   function displayHours(data){
9     if(!data.morning && !data.afternoon){
10      console.log("No se encontraron horas para el medico ese dia
     ");
11      $hours.html(noHoursAlert);
12      return;
13    }
14
15
```

Listing 31: Llamar ruta de intervalos de horas

Con las siguientes funciones retornamos los intervalos.

```
1   private function getIntervals($start, $end, $date, $doctorId){
2     $start = new Carbon($start);
3     $end = new Carbon($end);
4
5     $intervals = [];
6     while($start < $end){
7       $interval = [];
8
9       $interval['start'] = $start->format('g:i A');
10
11      // no existe una cita para esta hora con este medico
12      $available = $this->isAvailableInterval($date, $doctorId,
    $start);
13
14      $start->addMinutes(30);
15      $interval['end'] = $start->format('g:i A');
16
17      if($available){
18        $intervals [] = $interval;
19      }
20    }
21    return $intervals;
22  }
23
```

```
24
25
26      public function getAvailableIntervals($date, $doctorId)
27      {
28       $workDay = WorkDay::where('active', true)
29       ->where('day', $this->getDayFromDate($date))
30       ->where('user_id', $doctorId)
31       ->first([
32       'morning_start', 'morning_end',
33       'afternoon_start', 'afternoon_end'
34       ]);
35       if(!$workDay){
36         return [];
37       }
38       $morningIntervals = $this->getIntervals(
39       $workDay->morning_start, $workDay->morning_end, $date,
    $doctorId);
40       $afternoonIntervals = $this->getIntervals(
41       $workDay->afternoon_start, $workDay->afternoon_end
42       ,$date, $doctorId
43       );
44       $data = [];
45       $data['morning']= $morningIntervals;
46       $data['afternoon']= $afternoonIntervals;
47       return $data;
48      }
49
```

Listing 32: Retornar los intervalos

9. Desarrollo de la aplicación móvil

Al crear el proyecto no configura la pantalla principal activity-main y el main.java

9.1. Diseño de Loging

Como la vista es muy simple se va utilizar un LinerarLayout vertical para que nos coloque un elemento encima de otro. los atributos wrap-content significa que esta asociado al contendo y match-parent es que utilize el acho o alto total, tendremos un textView con el texto Inicio de sesion, despues dos texView con E-mail y contraseña, dos campos EditText para llenar, un boton para registra y finalmente un textView con un click asociado. la medidia que se usa en android no son px sino dp, pero para los textos se usa sp, puedo repartir pesos con weight.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res
/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical"
android:padding="8dp"
android:background="@drawable/backgroundsss"
tools:context=".MainActivity">

<TextView
android:layout_width="match_parent"
android:layout_height="0dp"
android:layout_weight="1"
android:textSize="30sp"
android:background="@color/transparent"
android:gravity="center_horizontal"
android:text="@string/label_login"
android:textColor="@color/letras"
android:textAppearance="@style/Base.TextAppearance.AppCompat.
Medium" />

<LinearLayout
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="@color/transparent"
android:padding="8dp"
android:orientation="vertical">

<TextView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:gravity="center_horizontal"
android:layout_marginTop="12dp"
android:textSize="20sp"
android:textColor="@color/letras"
android:text="@string/label_email" />
```

```
38
39      <EditText
40      android:inputType="textEmailAddress"
41      android:layout_width="match_parent"
42      android:layout_height="wrap_content"
43      android:layout_marginTop="12dp"
44      android:textColor="@color/letras"/>
45
46      <TextView
47      android:layout_width="match_parent"
48      android:layout_height="wrap_content"
49      android:layout_marginTop="12dp"
50      android:gravity="center_horizontal"
51      android:textColor="@color/letras"
52      android:textSize="20sp"
53      android:text="@string/label_password" />
54
55      <EditText
56      android:inputType="textPassword"
57      android:layout_width="match_parent"
58      android:layout_height="wrap_content"
59      android:layout_marginTop="12dp"
60      android:textColor="@color/letras"/>
61
62
63      <Button
64      android:layout_width="match_parent"
65      android:layout_height="wrap_content"
66      android:layout_marginTop="12dp"
67      android:text="@string/buton_login"
68      style="@style/Base.Widget.AppCompat.Button.Colored"/>
69
70      <TextView
71      android:layout_width="match_parent"
72      android:layout_height="wrap_content"
73      android:layout_marginTop="12dp"
74      android:layout_marginBottom="16dp"
75      android:gravity="center_horizontal"
76      android:textColor="@color/letras"
77      android:text="@string/go_to_registrer" />
78
79      </LinearLayout>
80
81
82      </LinearLayout>
83
```

Listing 33: Pantalla Login

9.2. Diseño del Registro

```
1       <?xml version="1.0" encoding="utf-8"?>
2       <LinearLayout xmlns:android="http://schemas.android.com/apk/res
        /android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       android:orientation="vertical"
8       tools:context=".MainActivity">
9
```

```
10      <TextView
11      android:paddingTop="12dp"
12      android:layout_width="match_parent"
13      android:layout_height="0dp"
14      android:layout_weight="1"
15      android:textSize="30sp"
16      android:background="@drawable/bg_gradient"
17      android:gravity="center_horizontal"
18      android:text="@string/label_regis"
19      android:textColor="@color/white"
20      android:textAppearance="@style/Base.TextAppearance.AppCompat.
        Medium" />
21
22      <LinearLayout
23      android:layout_width="match_parent"
24      android:layout_height="wrap_content"
25      android:background="@color/colorPrimaryDark"
26      android:padding="8dp"
27      android:orientation="vertical">
28
29      <TextView
30      android:layout_width="match_parent"
31      android:layout_height="wrap_content"
32      android:gravity="center_horizontal"
33      android:layout_marginTop="12dp"
34      android:textSize="20sp"
35      android:textColor="@color/white"
36      android:text="@string/label_name" />
37
38      <EditText
39      android:inputType="textEmailAddress"
40      android:layout_width="match_parent"
41      android:layout_height="wrap_content"
42      android:layout_marginTop="12dp"
43      android:textColor="@color/white"/>
44
45      <TextView
46      android:layout_width="match_parent"
47      android:layout_height="wrap_content"
48      android:gravity="center_horizontal"
49      android:layout_marginTop="12dp"
50      android:textSize="20sp"
51      android:textColor="@color/white"
52      android:text="@string/label_email" />
53
54      <EditText
55      android:inputType="textEmailAddress"
56      android:layout_width="match_parent"
57      android:layout_height="wrap_content"
58      android:layout_marginTop="12dp"
59      android:textColor="@color/white"/>
60
61      <TextView
62      android:layout_width="match_parent"
63      android:layout_height="wrap_content"
64      android:layout_marginTop="12dp"
65      android:gravity="center_horizontal"
66      android:textColor="@color/white"
67      android:textSize="20sp"
68      android:text="@string/label_password" />
69
70      <EditText
```

```
    android:inputType="textPassword"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="12dp"
    android:textColor="@color/white"/>


    <Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="12dp"
    android:text="@string/buton_register"
    style="@style/Base.Widget.AppCompat.Button.Colored"/>

    <TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="12dp"
    android:layout_marginBottom="16dp"
    android:gravity="center_horizontal"
    android:textColor="@color/white"
    android:text="@string/go_to_login" />

    </LinearLayout>


    </LinearLayout>

```

Listing 34: Pantalla de Registro

En Kotlin es muy sencillo ya no hay que hacer referencia a los id sino que podemos utilizarlos directament, por ejemplo si presionamos al id correspondiente podemos mandar llamar un TOAST.

```
    import android.os.Bundle
    import android.view.View
    import android.widget.TextView
    import android.widget.Toast
    import kotlinx.android.synthetic.main.activity_main.*

    class MainActivity : AppCompatActivity() {

      override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        tvGoToRegister.setOnClickListener {
          Toast.makeText(this, "Hola soy un Toast", Toast.
    LENGTH_SHORT)
        }


      }
    }

```

Listing 35: Toast con Kotlin

Para abrir otro activity desde uno se usa el metodo Intetn y starIntent

```
  import android.os.Bundle
  import android.view.View
  import android.widget.TextView
```

```
import android.widget.Toast
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity() {

  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    tvGoToRegister.setOnClickListener {
      Toast.makeText(this, getString(R.string.
  please_fill_your_data), Toast.LENGTH_SHORT)

      // si usamos var la variable puede cambiar pero si usamos
  val no cambiara
      // para no tener error tenemos que poner .java al final
  porque intent es una clase java
      val intent = Intent(this, RegisterActivity::class.java)
      startActivity(intent)
    }


  }
}
```

Listing 36: Intent entre pantallas

9.3. Diseño del Menú de opciones

La aplicación móvil tiene 3 opciones la primera es para crear una cíta, la segunda es para ver las citas y sus status y la tercera es para cerrar sessión.

```
  <?xml version="1.0" encoding="utf-8"?>
  <LinearLayout xmlns:android="http://schemas.android.com/apk/res
  /android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:gravity="center"
  android:padding="16dp"
  android:background="@drawable/bg_gradient"
  tools:context=".MenuActivity">

  <TextView
  android:paddingTop="12dp"
  android:layout_width="match_parent"
  android:layout_height="60dp"
  android:textSize="30sp"
  android:gravity="center_horizontal"
  android:text="@string/option_menu"
  android:textColor="@color/white"
  android:textAppearance="@style/Base.TextAppearance.AppCompat.
  Medium" />


```

```
24
25
26      <Button
27      android:layout_width="match_parent"
28      android:layout_marginTop="12dp"
29      android:drawableStart="@drawable/ic_add_circle"
30      android:text="@string/create_appointment"
31      style="@style/menuButton"/>
32
33      <Button
34      android:layout_width="match_parent"
35      android:layout_marginTop="12dp"
36      android:drawableStart="@drawable/ic_schedule"
37      android:text="@string/my_appointment"
38      style="@style/menuButton"/>
39
40      <Button
41      android:layout_width="match_parent"
42      android:layout_marginTop="12dp"
43      android:layout_marginBottom="16dp"
44      android:drawableStart="@drawable/ic_power_off"
45      android:text="@string/logout"
46      style="@style/menuButton"/>
47
48      </LinearLayout>
49
```

Listing 37: Diseño Menú de opciones

9.4. Diseño crear cita

Primero vamos a solicitar la especialidad del médico, un comentario, el tipo de cita, el médico y la fecha. primero tendremos un cardview visible y al presionar next lo haremos invisible y mostraremos el otro.

```
1       <?xml version="1.0" encoding="utf-8"?>
2       <LinearLayout xmlns:android="http://schemas.android.com/apk/res
        /android"
3       xmlns:app="http://schemas.android.com/apk/res-auto"
4       xmlns:tools="http://schemas.android.com/tools"
5       android:layout_width="match_parent"
6       android:layout_height="match_parent"
7       android:background="@drawable/bg_gradient"
8       android:gravity="center"
9       android:orientation="vertical"
10      android:padding="16dp"
11      tools:context=".CreateAppointmentActivity">
12
13      <TextView
14      android:layout_width="match_parent"
15      android:layout_height="60dp"
16      android:gravity="center_horizontal"
17      android:paddingTop="12dp"
18      android:text="@string/create_appointment"
19      android:textAppearance="@style/Base.TextAppearance.AppCompat.
        Medium"
20      android:textColor="@color/white"
21      android:textSize="30sp" />
22
23      <android.support.v7.widget.CardView
24      android:id="@+id/cvStep1"
```

```
25      android:layout_width="match_parent"
26      android:layout_height="wrap_content"
27      app:cardCornerRadius="8dp"
28      app:cardElevation="8dp"
29      app:contentPadding="12dp">
30
31      <LinearLayout
32      android:layout_width="match_parent"
33      android:layout_height="match_parent"
34      android:orientation="vertical">
35
36      <TextView
37      android:layout_width="match_parent"
38      android:layout_height="wrap_content"
39      android:layout_marginTop="8dp"
40      android:gravity="center_horizontal"
41      android:text="@string/label_descripcion"
42      android:textColor="@color/letras" />
43
44      <EditText
45      android:layout_width="match_parent"
46      android:layout_height="wrap_content"
47      android:textColor="@color/letras" />
48
49      <TextView
50      android:layout_width="match_parent"
51      android:layout_height="wrap_content"
52      android:layout_marginTop="8dp"
53      android:gravity="center_horizontal"
54      android:text="@string/label_specialty"
55      android:textColor="@color/letras" />
56
57      <Spinner
58      android:layout_width="match_parent"
59      android:layout_height="wrap_content"></Spinner>
60
61      <TextView
62      android:layout_width="match_parent"
63      android:layout_height="wrap_content"
64      android:layout_marginTop="8dp"
65      android:gravity="center_horizontal"
66      android:text="@string/label_tipo_cita"
67      android:textColor="@color/letras" />
68
69      <RadioGroup
70      android:layout_width="match_parent"
71      android:layout_height="wrap_content"
72      android:layout_marginBottom="8dp"
73      android:orientation="vertical">
74
75      <RadioButton
76      android:layout_width="wrap_content"
77      android:layout_height="wrap_content"
78      android:text="@string/tipo1"
79      android:textColor="@color/letras" />
80
81      <RadioButton
82      android:layout_width="wrap_content"
83      android:layout_height="wrap_content"
84      android:text="@string/tipo2"
85      android:textColor="@color/letras" />
86
```

```
87      <RadioButton
88      android:layout_width="wrap_content"
89      android:layout_height="wrap_content"
90      android:text="@string/tipo3"
91      android:textColor="@color/letras" />
92
93      </RadioGroup>
94
95      <Button
96      android:id="@+id/btnNext"
97      style="@style/menuButton"
98      android:layout_width="match_parent"
99      android:layout_marginTop="12dp"
100     android:layout_marginBottom="16dp"
101     android:drawableEnd="@drawable/ic_navigate_next"
102     android:text="@string/btn_next" />
103
104     </LinearLayout>
105
106     </android.support.v7.widget.CardView>
107
108
109     <android.support.v7.widget.CardView
110     android:id="@+id/cvStep2"
111     android:layout_width="match_parent"
112     android:layout_height="wrap_content"
113     android:visibility="gone"
114     app:cardCornerRadius="8dp"
115     app:cardElevation="8dp"
116     app:contentPadding="12dp">
117
118     <android.support.v4.widget.NestedScrollView
119     android:layout_width="match_parent"
120     android:layout_height="wrap_content">
121
122     <LinearLayout
123     android:layout_width="match_parent"
124     android:layout_height="match_parent"
125     android:orientation="vertical">
126
127     <TextView
128     android:layout_width="match_parent"
129     android:layout_height="wrap_content"
130     android:layout_marginTop="8dp"
131     android:gravity="center_horizontal"
132     android:text="@string/label_doctor"
133     android:textColor="@color/letras" />
134
135     <Spinner
136     android:layout_width="match_parent"
137     android:layout_height="wrap_content"></Spinner>
138
139     <TextView
140     android:layout_width="match_parent"
141     android:layout_height="wrap_content"
142     android:layout_marginTop="8dp"
143     android:gravity="center_horizontal"
144     android:text="@string/label_schedule_date"
145     android:textColor="@color/letras" />
146
147     <EditText
148     android:layout_width="match_parent"
```

```
149     android:layout_height="wrap_content"
150     android:textColor="@color/letras" />
151
152     <RadioGroup
153     android:layout_width="match_parent"
154     android:layout_height="wrap_content"
155     android:layout_marginBottom="8dp"
156     android:orientation="vertical">
157
158     <RadioButton
159     android:layout_width="wrap_content"
160     android:layout_height="wrap_content"
161     android:text="8:00 AM"
162     android:textColor="@color/letras" />
163
164     <RadioButton
165     android:layout_width="wrap_content"
166     android:layout_height="wrap_content"
167     android:text="8:30 AM"
168     android:textColor="@color/letras" />
169
170     <RadioButton
171     android:layout_width="wrap_content"
172     android:layout_height="wrap_content"
173     android:text="9:00 AM"
174     android:textColor="@color/letras" />
175
176     <RadioButton
177     android:layout_width="wrap_content"
178     android:layout_height="wrap_content"
179     android:text="8:00 AM"
180     android:textColor="@color/letras" />
181
182     <RadioButton
183     android:layout_width="wrap_content"
184     android:layout_height="wrap_content"
185     android:text="8:30 AM"
186     android:textColor="@color/letras" />
187
188     <RadioButton
189     android:layout_width="wrap_content"
190     android:layout_height="wrap_content"
191     android:text="9:00 AM"
192     android:textColor="@color/letras" />
193
194     <RadioButton
195     android:layout_width="wrap_content"
196     android:layout_height="wrap_content"
197     android:text="8:00 AM"
198     android:textColor="@color/letras" />
199
200     <RadioButton
201     android:layout_width="wrap_content"
202     android:layout_height="wrap_content"
203     android:text="8:30 AM"
204     android:textColor="@color/letras" />
205
206     <RadioButton
207     android:layout_width="wrap_content"
208     android:layout_height="wrap_content"
209     android:text="9:00 AM"
210     android:textColor="@color/letras" />
```

```
211
212     <RadioButton
213     android:layout_width="wrap_content"
214     android:layout_height="wrap_content"
215     android:text="8:00 AM"
216     android:textColor="@color/letras" />
217
218     <RadioButton
219     android:layout_width="wrap_content"
220     android:layout_height="wrap_content"
221     android:text="8:30 AM"
222     android:textColor="@color/letras" />
223
224     <RadioButton
225     android:layout_width="wrap_content"
226     android:layout_height="wrap_content"
227     android:text="9:00 AM"
228     android:textColor="@color/letras" />
229
230     <RadioButton
231     android:layout_width="wrap_content"
232     android:layout_height="wrap_content"
233     android:text="8:00 AM"
234     android:textColor="@color/letras" />
235
236     <RadioButton
237     android:layout_width="wrap_content"
238     android:layout_height="wrap_content"
239     android:text="8:30 AM"
240     android:textColor="@color/letras" />
241
242     <RadioButton
243     android:layout_width="wrap_content"
244     android:layout_height="wrap_content"
245     android:text="9:00 AM"
246     android:textColor="@color/letras" />
247
248     <RadioButton
249     android:layout_width="wrap_content"
250     android:layout_height="wrap_content"
251     android:text="8:00 AM"
252     android:textColor="@color/letras" />
253
254     <RadioButton
255     android:layout_width="wrap_content"
256     android:layout_height="wrap_content"
257     android:text="8:30 AM"
258     android:textColor="@color/letras" />
259
260     <RadioButton
261     android:layout_width="wrap_content"
262     android:layout_height="wrap_content"
263     android:text="9:00 AM"
264     android:textColor="@color/letras" />
265
266
267
268     </RadioGroup>
269
270
271     <Button
272     android:id="@+id/btnConfirmAppointment"
```

```
    style="@style/menuButton"
    android:layout_width="match_parent"
    android:layout_marginTop="12dp"
    android:layout_marginBottom="16dp"
    android:drawableEnd="@drawable/ic_check"
    android:text="@string/btn_confirm" />

    </LinearLayout>
    </android.support.v4.widget.NestedScrollView>

    </android.support.v7.widget.CardView>


    </LinearLayout>



    /******** CODIGO PARA OCULTAR CARDVIEWS ********/

    btnNext.setOnClickListener {
      cvStep1.visibility = View.GONE
      cvStep2.visibility = View.VISIBLE
    }

```

Listing 38: Diseño primer paso reservar cita

Para mayor funcionamiento poblaremos nuestro spinner con datos ficticios.

```
    // el material spinner arrayAdapter nos pide 3 parametros(
    contexto, de que forma se mostraran los datos
    // y finalmente el arreglo de datos)
    val specialtiesOptions = arrayOf("Specialty A","Specialty B","
    Specialty C")
    val adapter = ArrayAdapter<String>(this, android.R.layout.
    simple_list_item_1,specialtiesOptions)
    spinnerSpecialties.setAdapter(adapter)


    val doctorOptions = arrayOf("Medico 1","Medico 2","Medico 3")
    spinnerDoctors.adapter = ArrayAdapter<String>(this,android.R.
    layout.simple_list_item_1,doctorOptions)


```

Listing 39: Poblar spinner con datos de prueba

9.5. DatePicker Dialog

```
    /***************** XML******************/
     <EditText
    android:id="@+id/etScheduledDate"
    android:onClick="onClickScheduledDate"
    android:inputType="date"
    android:focusable="false"
    android:clickable="true"
    android:maxLines="1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/letras" />

    /************** KOTLIN *********************/
```

```
14    class CreateAppointmentActivity : AppCompatActivity() {
15      val calendar = Calendar.getInstance()
16
17
18    fun onClickScheduledDate(v: View){ // creo objeto calendario
19      // obtener el year, mes y dia del mes actual
20      // val calenadar = Calendar.getInstance()
21      val year = calendar.get(Calendar.YEAR)
22      val month = calendar.get(Calendar.MONTH)
23      val dayOfMonth = calendar.get(Calendar.DAY_OF_MONTH)
24
25      // Esto pasa cuando selecciono un dia //
26      val listener = DatePickerDialog.OnDateSetListener {
    datePicker, y, m, d ->
27        //Toast.makeText(this,"$y-$m-$d", Toast.LENGTH_SHORT).show
    ()
28        calendar.set(y,m,d)
29        etScheduledDate.setText(resources.getString(R.string.
    date_format, y, m, d))
30      }
31      // creo un dialogoPicker
32      DatePickerDialog(this, listener, year, month, dayOfMonth).
    show()
33    }
34
35
```

Listing 40: DatePicker XML y DateDIalog

9.6. Generar redioButton

```
1     /***************** XML*******************/
2
3   <RadioGroup
4   android:id="@+id/radioGroup"
5   android:layout_width="match_parent"
6   android:layout_height="wrap_content"
7   android:layout_marginBottom="8dp"
8   android:orientation="vertical">
9
10  /******* KOTLIN *********/
11   private fun displayRadioButtons(){
12    val radioButton = RadioButton(this)
13    radioButton.id = View.generateViewId()
14    radioButton.text = "RadioButton 1"
15    radioGroup.addView(radioButton)
16  }
17
```

Listing 41: Generar radioButtons

9.7. RadioGroup multiple columna

```
1     /***************** XML*******************/
2     <LinearLayout
3     android:layout_width="match_parent"
4     android:layout_height="wrap_content"
5     android:orientation="horizontal">
6
```

```
    <LinearLayout
    android:id="@+id/radioGroupLeft"
    android:layout_width="0dp"
    android:layout_weight="1"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:orientation="vertical">

    <RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="8:00 AM"
    android:textColor="@color/letras" />


    <RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="8:00 AM"
    android:textColor="@color/letras" />


    </LinearLayout>


    <LinearLayout
    android:id="@+id/radioGroupRight"
    android:layout_width="0dp"
    android:layout_weight="1"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:orientation="vertical">

    <RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="8:00 AM"
    android:textColor="@color/letras" />


    <RadioButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="8:00 AM"
    android:textColor="@color/letras" />


    </LinearLayout>

    </LinearLayout>


    /************** KOTLIN ********************/
    private fun displayRadioButtons(){
      // limpiar los radioButton ya existentes
      //radioGroup.clearCheck()
      selectedRadioButton = null
      // limpiar los elementos ya puestos
      radioGroupLeft.removeAllViews()
      radioGroupRight.removeAllViews()

      val hours = arrayOf("3:00 PM","3:30 PM","4:00 PM", "4:30 PM")
```

```
69      var goToLeft = true
70
71      hours.forEach {
72        val radioButton = RadioButton(this)
73        radioButton.id = View.generateViewId()
74        radioButton.text = it
75
76        // desmarcar los radio buton y marcar al que hizimos click
77        radioButton.setOnClickListener{ view->
78          selectedRadioButton?.isChecked = false
79          selectedRadioButton = view as RadioButton?
80          selectedRadioButton?.isChecked = true
81        }
82        if(goToLeft)
83        radioGroupLeft.addView(radioButton)
84        else
85        radioGroupRight.addView(radioButton)
86        goToLeft = !goToLeft
87      }
88
89      // obtener el id del radioButton Marcado
90      //radioGroup.checkedRadioButtonId
91
92    }
93
94
```

Listing 42: DatePicker XML y DateDIalog

10. Mis cítas en Android

Para la pantalla de mis citas implementaremos un RecyclerView cabe mencionar que para hacer esto es necesario implementar un adaptador y una clase de citas ademas una vista que complementara.

```
1       /***************** XML layout_appointments
    ******************/
2       <?xml version="1.0" encoding="utf-8"?>
3       <LinearLayout xmlns:android="http://schemas.android.com/apk/
    res/android"
4       xmlns:app="http://schemas.android.com/apk/res-auto"
5       xmlns:tools="http://schemas.android.com/tools"
6       android:layout_width="match_parent"
7       android:layout_height="match_parent"
8       android:orientation="vertical"
9       android:background="@drawable/bg_gradient"
10      tools:context=".AppointmentsActivity">
11
12      <TextView
13      android:layout_marginTop="12dp"
14      android:layout_width="match_parent"
15      android:layout_height="wrap_content"
16      android:textColor="@color/white"
17      style="@style/Base.TextAppearance.AppCompat.Medium"
18      android:gravity="center_horizontal"
19      android:text="@string/my_appointments"/>
20
21      <android.support.v7.widget.RecyclerView
22      android:id="@+id/rvAppointments"
23      tools:listItem="@layout/item_appointment"
```

```
24      android:padding="6dp"
25      android:layout_width="match_parent"
26      android:layout_height="match_parent">
27
28      </android.support.v7.widget.RecyclerView>
29
30      </LinearLayout>
31
32
33      /************************** Item_appointment
    *****************/
34      <?xml version="1.0" encoding="utf-8"?>
35      <android.support.v7.widget.CardView xmlns:android="http://
    schemas.android.com/apk/res/android"
36      xmlns:tools="http://schemas.android.com/tools"
37      android:layout_width="match_parent"
38      android:layout_height="wrap_content"
39      xmlns:app="http://schemas.android.com/apk/res-auto"
40      android:orientation="vertical"
41      android:gravity="center"
42      android:padding="16dp"
43      android:background="@drawable/bg_gradient"
44      app:cardUseCompatPadding="true"
45      app:cardElevation="4dp"
46      app:cardCornerRadius="3dp"
47      tools:context=".MenuActivity">
48
49      <LinearLayout
50      android:orientation="vertical"
51      android:padding="6dp"
52      android:layout_width="match_parent"
53      android:layout_height="wrap_content">
54
55      <TextView
56      android:layout_width="match_parent"
57      android:layout_height="wrap_content"
58      android:id="@+id/tvAppointmentId"
59      tools:text="Cita Medica # 7"/>
60
61      <TextView
62      android:layout_width="match_parent"
63      android:layout_height="wrap_content"
64      android:id="@+id/tvDoctorName"
65      tools:text="Medico Test"
66      android:gravity="center_horizontal"
67      style="@style/Base.TextAppearance.AppCompat.Medium"/>
68
69      <TextView
70      android:layout_width="match_parent"
71      android:layout_height="wrap_content"
72      android:gravity="center_horizontal"
73      android:id="@+id/tvScheduledDate"
74      tools:text="Atencion el dia 12/12/2021"/>
75
76      <TextView
77      android:layout_width="match_parent"
78      android:layout_height="wrap_content"
79      android:gravity="center_horizontal"
80      android:id="@+id/tvScheduledTime"
81      tools:text="A las 3:00 PM"/>
82
83
```

```
84
85
86        </LinearLayout>
87
88
89
90        </android.support.v7.widget.CardView>
91
92
93
94        /***************** ACTIVITY MY_APPOINTMENTS *************/
95        package com.lopez.samuel.myappointment
96
97        import android.support.v7.app.AppCompatActivity
98        import android.os.Bundle
99        import android.support.v7.widget.LinearLayoutManager
100       import com.lopez.samuel.myappointment.model.Appointment
101       import kotlinx.android.synthetic.main.activity_appointments.*
102
103       class AppointmentsActivity : AppCompatActivity() {
104
105         override fun onCreate(savedInstanceState: Bundle?) {
106           super.onCreate(savedInstanceState)
107           setContentView(R.layout.activity_appointments)
108
109
110           rvAppointments.layoutManager = LinearLayoutManager(this)
    // va ser lineal // gridLinealManager
111
112           val appointments = ArrayList<Appointment>()
113           appointments.add(Appointment(1,"Medico Test
    ","12/12/2021","4:00 PM"))
114           appointments.add(Appointment(2,"Medico Test
    ","12/12/2021","4:00 PM"))
115           appointments.add(Appointment(3,"Medico Test
    ","12/12/2021","4:00 PM"))
116
117           rvAppointments.adapter = AppointmentAdapter(appointments)
118         }
119       }
120
121
122
123       /*************** ADAPTADOR APPOINTMENTS
    ********************/
124
125       package com.lopez.samuel.myappointment
126
127       import android.support.v7.widget.RecyclerView
128       import android.view.LayoutInflater
129       import android.view.View
130       import android.view.ViewGroup
131       import com.lopez.samuel.myappointment.model.Appointment
132       import kotlinx.android.synthetic.main.item_appointment.view.*
133
134       class AppointmentAdapter(private val appointments: ArrayList<
    Appointment>): RecyclerView.Adapter<AppointmentAdapter.
    ViewHolder>(){
135
136         // representa nuestra vista
137         class ViewHolder(itemView: View): RecyclerView.ViewHolder(
    itemView) {
```

```
138
139         fun bind(appointment: Appointment) =
140         // estamos accediendo solo al objeto itemView
141         with(itemView){
142           tvAppointmentId.text = context.getString(R.string.
    item_appointment_id,appointment.id)
143           tvDoctorName.text = appointment.doctorName
144           tvScheduledDate.text = context.getString(R.string.
    item_appointment_date,appointment.scheduledDate)
145           tvScheduledTime.text = context.getString(R.string.
    item_appointment_time,appointment.scheduledTime)
146         }
147
148       }
149
150       // inflate XML items
151       override fun onCreateViewHolder(parent: ViewGroup, viewType
    : Int): ViewHolder {
152         return ViewHolder(
153         LayoutInflater.from(parent.context).inflate(R.layout.
    item_appointment, parent, false)
154         )
155       }
156
157       // Number of Elements
158       override fun getItemCount() = appointments.size
159
160
161       // Bind to fate with data
162       override fun onBindViewHolder(holder: ViewHolder, position:
     Int) {
163         val appointment = appointments[position]
164
165         holder.bind(appointment)
166
167       }
168     }
169
170
171   /******************** MODELO APPOINTMENT*******************/
172   package com.lopez.samuel.myappointment.model
173
174   data class Appointment (
175   val id: Int,
176   val doctorName: String,
177   val scheduledDate: String,
178   val scheduledTime: String
179
180   )
181
182
```

Listing 43: RecyclerView Citas

11. Navigation entre pantallas

En esta parte se tratara todos los apectos para que la aplicación sea consistente, empezando por los retornos o botones de anterior. Al asignarle un padre automaticamente se pone visible un boton de volver al padre

11.1. BackButton

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/
android"
package="com.lopez.samuel.myappointment">

<application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="@string/app_name"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"
android:theme="@style/AppTheme">
<activity android:name=".AppointmentsActivity"
android:parentActivityName=".MenuActivity"></activity>

<activity android:name=".CreateAppointmentActivity"
android:parentActivityName=".MenuActivity"/>

<activity android:name=".MenuActivity" />
<activity android:name=".RegisterActivity" />
<activity android:name=".MainActivity">
<intent-filter>
<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>

</manifest>

```

Listing 44: Back boton mediante parent

11.2. Una sola actividad y cerrar la actual cuando abrimos otra

Para cerrar la activity actual se pone finish() despues del startActivity(intent) y para que solo cree una instancia se necesita meter en el manifest en cada actividad se pone android:launchMode=”singleInstance”.

11.3. Persistencia de datos SharedPreference

```
val preferences = getSharedPreferences("general", Context.
MODE_PRIVATE)
val session = preferences.getBoolean("session",false)

if(session)
goToMenuActivity()

btnLogin.setOnClickListener {
// validar en servidor
createSessionPreference()
goToMenuActivity()
}


```

```
14      private fun createSessionPreference(){
15        val preferences = getSharedPreferences("general", Context.
    MODE_PRIVATE)
16        val editor = preferences.edit()
17        editor.putBoolean("session",true)
18        editor.apply()
19
20      }
21
22
23
24
25
26    btnLogout.setOnClickListener {
27      clearSessionPreference()
28      val intent = Intent(this,MainActivity::class.java)
29      startActivity(intent)
30      finish()
31
32    }
33
34
35   private fun clearSessionPreference(){
36    val preferences = getSharedPreferences("general", Context.
    MODE_PRIVATE)
37    val editor = preferences.edit()
38    editor.putBoolean("session",false)
39    editor.apply()
40
41  }
42
```

Listing 45: Shared Prefrence login y logout

11.4. Mejora del SharedPreference

Se utilizara un object en vez de un class estó da una diferencia de ya no crear clase si no usarlo como metodo estatico, pero tampoco contiene un constructor.

```
1       /***************** CLASE PREFERENT ******************/
2       package com.lopez.samuel.myappointment
3
4
5       import android.content.Context
6       import android.content.SharedPreferences
7       import android.preference.PreferenceManager
8
9       object PreferenceHelper {
10
11        fun defaultPrefs(context: Context): SharedPreferences
12        = PreferenceManager.getDefaultSharedPreferences(context)
13
14        fun customPrefs(context: Context, name: String):
    SharedPreferences
15        = context.getSharedPreferences(name, Context.MODE_PRIVATE)
16
17        private inline fun SharedPreferences.edit(operation: (
    SharedPreferences.Editor) -> Unit) {
18          val editor = this.edit()
19          operation(editor)
20          editor.apply()
```

```
21          }
22
23          /**
24          * puts a value for the given [key].
25          */
26          operator fun SharedPreferences.set(key: String, value: Any
    ?)
27          = when (value) {
28            is String? -> edit { it.putString(key, value) }
29            is Int -> edit { it.putInt(key, value) }
30            is Boolean -> edit { it.putBoolean(key, value) }
31            is Float -> edit { it.putFloat(key, value) }
32            is Long -> edit { it.putLong(key, value) }
33            else -> throw UnsupportedOperationException("Not yet
    implemented")
34          }
35
36          /**
37          * finds a preference based on the given [key].
38          * [T] is the type of value
39          * @param defaultValue optional defaultValue - will take a
    default defaultValue if it is not specified
40          */
41          inline operator fun <reified T : Any> SharedPreferences.get
    (key: String, defaultValue: T? = null): T
42          = when (T::class) {
43            String::class -> getString(key, defaultValue as? String
    ?: "") as T
44            Int::class -> getInt(key, defaultValue as? Int ?: -1) as
    T
45            Boolean::class -> getBoolean(key, defaultValue as?
    Boolean ?: false) as T
46            Float::class -> getFloat(key, defaultValue as? Float ?:
    -1f) as T
47            Long::class -> getLong(key, defaultValue as? Long ?: -1)
    as T
48            else -> throw UnsupportedOperationException("Not yet
    implemented")
49          }
50        }
51
52
53
54
55      /********************** CLASE MAIN ********************/
56      val preferences = PreferenceHelper.defaultPrefs(this) // va ser
         por default
57
58      if(preferences["session",false])
59      goToMenuActivity()
60
61
62       private fun createSessionPreference(){
63        /*
64        val preferences = getSharedPreferences("general", Context.
    MODE_PRIVATE)
65        val editor = preferences.edit()
66        editor.putBoolean("session",true)
67        editor.apply()
68        */
69        val preferences = PreferenceHelper.defaultPrefs(this)
70        preferences["session"] = true
```

```
71
72      }
73
74
75
76      private fun clearSessionPreference(){
77        /*val preferences = getSharedPreferences("general", Context.
        MODE_PRIVATE)
78        val editor = preferences.edit()
79        editor.putBoolean("session",false)
80        editor.apply()*/
81
82        val preferences = PreferenceHelper.defaultPrefs(this)
83        preferences["session"] = false
84
85
86      }
87
```

Listing 46: Mejora

11.5. Snackbar by Lazy

Para esto debemos sobrescribir el metodo onBackPress() que sirve para cuando presionamos la tecla de ir para atras, cada que se presione ese boton se va a mostrar un snackbar para confirmar la salida, para hacer uso de la clase Snackbar es necesario añadir una dependencia "design", para generar un snackbar se tienen que pasar tres parametros, el primero es la vista, el segundo es el texto y el ultimo el tiempo que se ejecutara, tambien se le tiene que dar un id a el layout donde se va a mostrar, al presionar el metdo onBackPress y ver que el snackbar ya esta visible entonces se sale de la app.

```
1     private val snackbar = SnackBar(vista, texto, duracion)
2
3     overrive fun onBackPressed(){
4       if(snackbar.isShow()){
5         super.onBackPressed()
6       }else{
7         snackbar.show()
8       }
9     }
10
11
```

Listing 47: SnackBar by Lazy

11.6. Autenticación JWT

Encoding consiste en transformar datos de un formato a otro, no busca mantener la información secreta se puede revertir facilmente no requiere de una clave, la encriptación tambien transofrma datos en otro formato pero la diferencia es que no cualuquiera puede regresar esa información a su valor original, la meta es que los datos no sean consumidos por otros usuarios a los que no fue destinada, tiene una clave secreta, El hasing no hay una encriptación y una desencriptación, solo es identifcar si ha sifo alterado y que el destinatario sepa si fue modificado, la entrada es la misma siemore va ser el mismo resultado, no

es reversible. Tambien podemos ofuscar el codigo que significa que las variables no describen lo que hacen y no tienen saltos de linea.

Hay muchas maneras de autenticar, una manera es la autentificación basada en tokens, en el servidor no se guardar ningun token simplemente se asegurara que sea valido el token, simplemente se firma y atravez de esta firma el servidor verifica si es valido y fue emitido poe el.

El usuario ingresa sus credenciales, el servidor verifica si son correctas y genera un token firmado (jwt), este token puede contener los datos en su interior, ese token se guarda del lado del cliente (preferencias en android), para pedir información privada se pide el token. se puede mandar mediante un Autoriztion Header,

El token esta compuesto de un Header, un payload y un asignature,paso 1.- header.payload.asignature, el header contiene el tipo y el algoritmo, 2.-payload contiene cleans, son datos que contiene información, iss nombre o dominio, sub descripcion exp cuando va expirar el token, 3.- crear la firma, se codifica el header y el payload con un algoritmo base64 y se concatnan, luego a ese resultado se aplica un algoritmo de hasing y con un secret, despues se vuelve a hasiar y se obtiene la firma, se firma en el servidor porque el tiene una llave privada, 4.- juntar todo el header el payload y la firma.

11.6.1. JWT con Passport

Client ID: 1 Client secret: AXBsF1DQEVYebFYqibVEMIYXDOGqIJ3i36Q2Dl47
Password grant client created successfully. Client ID: 2 Client secret: NIhVFJ418CP7ynImVfPK64RhgdhSqk7r

12. Retrofit con Kotlin

Para instalar Kotlin fue necesario copiar y pegar en el build.grade la siguiente dependencia es implementation 'com.squareup.retrofit2:retrofit:2.3.0' pero tambien tiene otra dependencia que ayuda a convertir los objetos Json que vienen del servidor a Objetos Kotlin automaticamente esa dependencia es implementation 'com.squareup.retrofit2:converter-gson:2.3.0', tambien es necesario darle permisos a la aplicación de internet este permiso se pone en el manifest uses-permission android:name='android.permission.INTERNET'. Primero vamos a crear una interfaz kotlin para usar la dependencia Retrofit. usamos una retorno Call para obtener los resultados del Json, usamos object Factory en vez de class Factory para que sea utilizado como statico y no sea necesario utilizar instancias. la ruta specialties retorna la lista de specialties registradas en la base de datos por ello se retorna call. Para que mi servidor corra con la ip le pongo php artisan serve –host=192.168.1.66

```
1   import retrofit2.Call
2   import retrofit2.Retrofit
3   import retrofit2.converter.gson.GsonConverterFactory
4   import retrofit2.http.GET
5
6   interface ApiService {
7
8     @GET("specialties")
9     abstract fun getSpecialties(): Call<ArrayList<Specialty>>
10
11
```

```
12      companion object Factory{
13        private const val BASE_URL = "http://192.168.1.66:800/api/"
14
15        fun create(): ApiService {
16          val retrofit = Retrofit.Builder()
17          .baseUrl(BASE_URL)
18          .addConverterFactory(GsonConverterFactory.create())
19          .build()
20          return retrofit.create(ApiService::class.java)
21        }
22      }
23    }
24
25
26    /********************** CLASE SPECIALTY ******************/
27    package com.lopez.samuel.myappointment.model
28
29    data class Specialty(val id: Int, val name: String) {
30      override fun toString(): String {
31        return name
32      }
33    }
34
35    /********************** CLASE Appointment ******************/
36    package com.lopez.samuel.myappointment.model
37
38    data class Appointment (
39    val id: Int,
40    val doctorName: String,
41    val scheduledDate: String,
42    val scheduledTime: String
43
44    )
45
```

Listing 48: Interfaz ApiService

Primero llamaeros las pespecialidades con el siguiente codigo. perimero hare una instancia de apiService para que solo se utilize una vez con un lazy o una carga presoza.

```
1       private val apiService: ApiService by lazy{
2       ApiService.create()
3     }
4
5
6     /************** CARGAR SPINNER DE ESPECIALIDADES
      ************************/
7     private fun loadSpecialties(){
8       val call = apiService.getSpecialties()
9       call.enqueue(object: Callback<ArrayList<Specialty>>{
10        override fun onFailure(call: Call<ArrayList<Specialty>>, t:
       Throwable) {
11          Toast.makeText(this@CreateAppointmentActivity,getString(R
      .string.error_loading_speialties), Toast.LENGTH_SHORT).show()
12          finish()
13        }
14
15        override fun onResponse(call: Call<ArrayList<Specialty>>,
      response: Response<ArrayList<Specialty>>) {
16          if(response.isSuccessful){ // cuando es correcto esta
      entre 200 y 300
```

```
17            val specialties = response.body() // obtenemos un
    arrayList de especialidades
18
19            val specialyOptions = ArrayList<String>()
20            specialties?.forEach{
21              specialyOptions.add(it.name)
22            }
23            spinnerSpecialties.adapter = ArrayAdapter<String>(
    this@CreateAppointmentActivity, android.R.layout.
    simple_list_item_1,specialyOptions)
24          }
25        }
26
27      })
28
29
30    }
31
```

Listing 49: Spinner de especialidades

Ahora mandaremos llamar a los médicos de acuerdo a una especialidad en este caso utilizremos un clickSelecteItem para cuando se haga un modificacion del spinner specialidades entonces se mande a llamar la ruta y asi llenamos el spinner de medicos. Pero tambien vamos a definir que el metodo toString de especialidades solo muestre el nombre.

```
1     private val apiService: ApiService by lazy{
2       ApiService.create()
3     }
4
5
6     /************** API Medicos segun la especialidad
      ***************************/
7     @GET("specialties/{specialty}/doctors")
8     abstract fun getDoctors(@Path("specialty") specialtyId: Int):
      Call<ArrayList<Doctor>>
9
10
11
12    /************** Esperar cambios del spinner specialties
      ************************/
13    private fun listenSpecialtyCHange(){
14      // obtengo el id del spinner Specialties
15      spinnerSpecialties.onItemSelectedListener = object:
      AdapterView.OnItemSelectedListener{
16        override fun onNothingSelected(parent: AdapterView<*>?) {
17
18        }
19
20        override fun onItemSelected(adapter: AdapterView<*>?, view:
       View?, position: Int, id: Long) {
21          // adapter nos permitira ingresar a los id
22          val specialty = adapter?.getItemAtPosition(position) as
      Specialty
23          Toast.makeText(this@CreateAppointmentActivity, "Id: ${
      specialty.id}", Toast.LENGTH_SHORT).show()
24          loadDoctors(specialty.id)
25        }
26
27      }
28    }
```

```
29
30
31  /****************************** CARGAR SPINNER DOCTOR
      *****************/
32  private fun loadDoctors(SpecialtyId: Int){
33    val call = apiService.getDoctors(SpecialtyId)
34    call.enqueue(object: Callback<ArrayList<Doctor>>{
35      override fun onFailure(call: Call<ArrayList<Doctor>>, t:
      Throwable) {
36          Toast.makeText(this@CreateAppointmentActivity,getString(R.
      string.error_loading_doctors), Toast.LENGTH_SHORT).show()
37
38      }
39
40      override fun onResponse(call: Call<ArrayList<Doctor>>,
      response: Response<ArrayList<Doctor>>) {
41          if(response.isSuccessful){ // cuando es correcto esta entre
       200 y 300
42            val doctors = response.body() // obtenemos un arrayList
      de especialidades
43
44            spinnerDoctors.adapter = ArrayAdapter<Doctor>(
      this@CreateAppointmentActivity, android.R.layout.
      simple_list_item_1,doctors)
45          }
46      }
47
48    })
49
50  }
51
52
```

Listing 50: Spinner de medicos

Para cargar las horas es necesario seleccionar una fecha y un médico y saldran las horas en las que puede atender dicho médico.

```
1     /****************** API HORAS
      ************************************/
2     @GET("schedule/hours")
3     abstract fun getHours(@Query("doctor_id") doctorId: Int, @Query
      ("date") date: String): Call<Schedule>
4
5
6     /****************** ESPERAR CAMBIOS DEL SPINNER MEDICO Y FECHA
      ***********/
7        private fun listenDoctorAndDateChange(){
8       // medicos
9       spinnerDoctors.onItemSelectedListener = object: AdapterView.
      OnItemSelectedListener{
10          override fun onNothingSelected(parent: AdapterView<*>?) {
11            // en caso de no tener ningun elemento seleccionado
12          }
13
14          override fun onItemSelected(adapter: AdapterView<*>?, view:
       View?, position: Int, id: Long) {
15            val doctor = adapter?.getItemAtPosition(position) as
      Doctor
16            loadHours(doctor.id, etScheduledDate.text.toString())
17          }
18
19      }
```

```
20
21      // fechas
22      etScheduledDate.addTextChangedListener(object: TextWatcher{
23        override fun afterTextChanged(s: Editable?) {
24          // DESPUES DE QUE EL TEXTO CAMBIO
25        }
26
27        override fun beforeTextChanged(s: CharSequence?, start: Int
    , count: Int, after: Int) {
28          // ANTES DE QUE EL TEXTO CAMBIO
29        }
30
31        override fun onTextChanged(s: CharSequence?, start: Int,
    before: Int, count: Int) {
32          // CUANDO EL TEXTO CAMBIO
33          val doctor = spinnerDoctors.selectedItem as Doctor
34          loadHours(doctor.id, etScheduledDate.text.toString())
35        }
36
37      })
38
39    }
40
41
42
43  /********************** FUNCION CARGAR HORAS ****************/
44  private fun loadHours(doctorId: Int, date: String){
45    if(date.isEmpty()){
46      return
47    }
48
49    val call = apiService.getHours(doctorId,date)
50    call.enqueue(object: Callback<Schedule>{
51      override fun onFailure(call: Call<Schedule>, t: Throwable) {
52        Toast.makeText(this@CreateAppointmentActivity, getString(R.
    string.error_loading_hours), Toast.LENGTH_SHORT).show()
53
54      }
55
56      override fun onResponse(call: Call<Schedule>, response:
    Response<Schedule>) {
57        if(response.isSuccessful){
58          val schedule = response.body()
59          schedule?.let{
60
61            tvSelectDoctorAndDate.visibility = View.GONE
62            val intervals = it.morning + it.afternoon
63            val hours = ArrayList<String>()
64            intervals.forEach{interval ->
65              hours.add(interval.start)
66            }
67            displayRadioButtons(hours)
68          }
69
70          //Toast.makeText(this@CreateAppointmentActivity,"Morning:
     ${schedule?.morning?.size} - Afternoon: ${schedule?.afternoon
    ?.size}", Toast.LENGTH_SHORT).show()
71        }
72      }
73
74    })
75  }
```

```
76
77
78  /************************ MODELOS ******************/
79  package com.lopez.samuel.myappointment.model
80
81  data class Schedule(val morning: ArrayList<HourInterval>, val
      afternoon: ArrayList<HourInterval>)
82
83
84  package com.lopez.samuel.myappointment.model
85
86  data class HourInterval(val start: String, val end: String){
87    override fun toString(): String {
88      return "$start - $end"
89    }
90  }
91
```

Listing 51: Desplegar horas