

Desarrollo de un sistema web para reservas de citas

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 usuarios (Administrador, Doctor y Paciente). El administrador que es el usuario más abstracto es el encargado de llenar los catálogos fuertes o catálogos principales que son el de (especialidad, médicos, pacientes) pero también puede aceptar o rechazar citas de todos los médicos y pacientes además puede ver las citas de todos los médicos y pacientes y es capaz de cancelarlas si es necesario, las citas tienen tres estados principales que son (reservada, confirmada y cancelada).

Además el usuario administrador tiene la posibilidad de ver en un gráfico la frecuencia de citas por día, mes etc y también los médicos más activos.

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

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

El usuario paciente también 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 específica 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 autenticación y registro automático se ingresa el siguiente comando `"php artisan make:auth"`. Automáticamente nos crea la migración `user` y también la tabla `password-reset` y también 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 diferente de acuerdo 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','
4     SpecialtyController@edit');
5 Route::post('/specialties','SpecialtyController@store');
6 Route::put('/specialties/{specialty}','
7     SpecialtyController@update');
8 Route::delete('/specialties/{specialty}','
9     SpecialtyController@destroy');
```

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.

```

1 <form action="{{ url('specialties') }}" method="POST">
2 @csrf
3 <div class="form-group">
4 <label for="name">Nombre de la especialidad</label>
5 <input type="text" name="name" class="form-control" value="{{
6 old('name') }}" required>
7 </div>
8 <div class="form-group">
9 <label for="name">Descripcion</label>
10 <input type="text" name="description" class="form-control">
11 </div>
12 <button type="submit" class="btn btn-primary">Guardar</button>
13 </form>
14

```

Listing 3: Formulario Especialidad

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

```

1 public function store(Request $request){
2     //dd($request->all());
3     $this->performValidation($request);
4     // si hay un error lo retorna y bloquea el flujo
5
6     $specialty = new Specialty();
7     $specialty->name = $request->input('name');
8     $specialty->description = $request->input('description');
9     $specialty->save(); // realiza un insert en la tabla
10    specialties
11
12    $notificacion = 'La especialidad se ha registrado
13    correctamente';
14    return redirect('/specialties')->with(compact('notificacion'))
15    }; // la variable se va a crear como una variable de sesion
16 }
17

```

Listing 4: Metodo para crear especialidad

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

```

1 public function index(){
2     // obtener el listado de especialidades
3     $specialties = Specialty::all();
4     return view('specialties.index',compact('specialties'));
5 }
6

```

Listing 5: obtener toda la lista de especialidades

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

```

1 @foreach($specialties as $specialty)
2 <tr>
3 <th scope="row">
4 {{ $specialty->name }}
5 </th>
6 <td>
7 {{ $specialty->description }}
8 </td>

```

```

9      <td>
10
11      <form action="{ url('specialties/'. $specialty->id) }}" method
12      ="POST">
13      @csrf
14      @method('DELETE')
15      <a href="{ url('specialties/'. $specialty->id.'/edit') }}"
16      class="btn btn-sm btn-primary">Editar</a>
17      <button type="submit" class="btn btn-sm btn-danger">Eliminar</
18      button>
19      </form>
20      </td>
21      </tr>
22      @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.

```

1      $rules = [
2      'name' => 'required|min:3'
3      ];
4      $messages = [
5      'name.required' => 'Es necesario ingresar el nombre ',
6      'name.min' => 'Como minimo el nombre debe tener 3 caracteres'
7      ];
8      $this->validate($request,$rules, $messages);
9
10     /***** MOSTRAR LOS ERRORES *****/
11     <div class="card-body">
12     @if ($errors->any())
13     <div class="alert alert-danger" role="alert">
14     <ul>
15     @foreach($errors->all() as $error)
16     <li>{{ $error }}</li>
17     @endforeach
18     </ul>
19     </div>
20     @endif
21

```

Listing 7: Validaciones del lado servidor

4.1. Actualizar datos de la especialidad

Primero mandamos llamar la ruta del método edit para que nos devuelva la especialidad seleccionada y nos permita pintar los datos en la vista, se observará 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.

```

1      public function edit(Specialty $specialty){
2      return view('specialties.edit', compact('specialty'));
3      }

```

```

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 $specialty)
2      {
3          //dd($request->all());
4          $this->performValidation($request);
5
6          $specialty->name = $request->input('name');
7          $specialty->description = $request->input('description');
8          $specialty->save(); // realiza un UPDATE en la tabla
          specialties
9          $notificacion = 'La especialidad se ha actualizado
          correctamente';
10         return redirect('/specialties')->with(compact('notificacion'))
11     };
12 }

```

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 usuarios

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/UserFactory, 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$92IXUNpkj00r0Q5byMi.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

Después 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 siguiente 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 en cuenta 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 = [

```

```

16     'name' => 'required|min:3',
17     'email' => 'required|email',
18     'dni' => 'nullable|digits:8',
19     'address' => 'nullable|min:5',
20     'phone' => 'nullable|min:6'
21 ];
22 $this->validate($request,$rules);
23
24     $user = User::create(
25     $request->only('name','email','dni','address','phone')
26     + [
27     'role' => 'doctor',
28     'password' => bcrypt($request->input('password'))
29     ]
30     );
31 }
32
33
34
35     $notificacion = 'El Medico se ha registrado correctamnete.';
36     return redirect('/doctors')->with(compact('notificacion'));
37

```

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.

```

1 public function edit($id)
2 {
3     $doctor = User::doctors()->findOrFail($id);
4     return view('doctors.edit',compact('doctor'));
5 }
6

```

Listing 17: Buscar doctor

La contraseña no siempre 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.

```

1 public function update(Request $request, $id)
2 {
3     $rules = [
4     'name' => 'required|min:3',
5     'email' => 'required|email',
6     'dni' => 'nullable|digits:8',
7     'address' => 'nullable|min:5',
8     'phone' => 'nullable|min:6'
9     ];
10    $this->validate($request,$rules);
11
12    $user = User::doctors()->findOrFail($id); // busco el objeto
13    $data = $request->only('name','email','dni','address','phone
14    ');
15    $password = $request->input('password');
16    if($password)
17    $data['password'] = bcrypt($password);
18
19    $user->fill($data);
20    $user->save();
21
22    $user->specialties()->sync($request->input('specialties'));

```

```

22         $notificacion = 'El Medico se ha actualizado correctamnete.';
23         return redirect('/doctors')->with(compact('notificacion'));
24     }
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(
11        function(){
12            Route::get('/specialties','SpecialtyController@index');
13            Route::get('/specialties/create','SpecialtyController@create');
14            Route::get('/specialties/{specialty}/edit','SpecialtyController@edit');
15
16            Route::post('/specialties','SpecialtyController@store');
17            Route::put('/specialties/{specialty}','SpecialtyController@update');
18            Route::delete('/specialties/{specialty}','SpecialtyController@destroy');
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.

```

1  protected $routeMiddleware = [
2      'auth' => \App\Http\Middleware\Authenticate::class,
3      'auth.basic' => \Illuminate\Auth\Middleware\
4      AuthenticateWithBasicAuth::class,
5      'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings
6      ::class,
7      'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders
8      ::class,
9      'can' => \Illuminate\Auth\Middleware\Authorize::class,
10     'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
11     'password.confirm' => \Illuminate\Auth\Middleware\
12     RequirePassword::class,
13     'signed' => \Illuminate\Routing\Middleware\ValidateSignature::
14     class,
15     'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::
16     class,
17     'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified
18     ::class,
19     'admin' => \App\Http\Middleware\AdminMiddleware::class,
20     'doctor' => \App\Http\Middleware\DoctorMiddleware::class,
21 ];

```

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.

```

1  /***** DEFINIR RUTAS *****/
2
3  Route::middleware(['auth', 'doctor'])->namespace('Doctor')->group(
4      function(){
5          Route::get('/schedule', 'ScheduleController@edit');
6          Route::post('/schedule', 'ScheduleController@store');
7      });
8

```

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).

```

1  public function up()
2  {
3      Schema::create('work_days', function (Blueprint $table) {
4          $table->id();
5          $table->unsignedSmallInteger('day');
6          $table->boolean('active');
7
8          $table->time('morning_start');
9          $table->time('morning_end');
10
11         $table->time('afternoon_start');
12         $table->time('afternoon_end');
13     });

```

```

14         $table->unsignedBigInteger('user_id');
15         $table->foreign('user_id')->references('id')->on('users');
16
17         $table->timestamps();
18     });
19 }
20

```

Listing 22: ScheduleController edit

Primero creo un arreglo de los 7 días ['lunes','martes',.....], pero tambien vamos a buscar los días de trabajo de acuerdo 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 días. Tambien cabe señalar que utilizamos el metodo in-array que busca un elemento dentro de un array.

```

1     private $days = ['Lunes','Martes','Miercoles','Jueves','Viernes',
2         'Sabado','Domingo'];
3     public function edit(){
4
5         $workDays = WorkDay::where('user_id',auth()->user()->id)->get();
6         // la siguiente funcion me permite recibir arreglo y
7         convertirlo a otro
8         if(count($workDays) > 0){
9             $workDays->map(function($workDay){
10                 $workDay->morning_start = (new Carbon($workDay->
11                     morning_start))->format('g:i A');
12                 $workDay->morning_end = (new Carbon($workDay->morning_end
13                     ))->format('g:i A');
14                 $workDay->afternoon_start = (new Carbon($workDay->
15                     afternoon_start))->format('g:i A');
16                 $workDay->afternoon_end = (new Carbon($workDay->
17                     afternoon_end))->format('g:i A');
18                 return $workDay;
19             });
20         }else{
21             $workDays = collect();
22             for($i=0; $i<7; $i++){
23                 $workDays->push(new WorkDay());
24             }
25             $days = $this->days;
26             return view('schedule',compact('workDays','days'));
27         }
28     }
29
30     /***** EN LA VISTA SE MUESTRAN *****/
31
32     <table class="table align-items-center table-flush">
33     <thead class="thead-light">
34     <tr>
35     <th scope="col">Dias</th>
36     <th scope="col">Activo</th>
37     <th scope="col">Turno Manana</th>
38     <th scope="col">Turno Tarde</th>
39     </tr>
40     </thead>
41     <tbody>
42     @foreach($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 llegarían 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 registro con los datos del segundo arreglo y si si estan solo actualiza 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 mañana son
      inconsistentes 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
      inconsistentes 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 necesario 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          $table->timestamps();
32      });
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 especialidad 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 tendríamos que indicarle a laravel el nombre de a tabla que queremos usar con `protected table = 'nombre de tabla'`, tambien hay una convención para las llaves 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      {
5          $table->id();
6          // doctor
7          $table->unsignedBigInteger('user_id');
8          $table->foreign('user_id')->references('id')->on('users');
9          // especialidad
10         $table->unsignedBigInteger('specialty_id');
11         $table->foreign('specialty_id')->references('id')->on('
specialties');
12
13         $table->timestamps();
14     });
15 }
16
17 /***** MODELO USER *****/
18
19 public function specialties()
20 {

```

```

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 especialidades para un médico necesitamos usar el bootstrap-select, que es un plugin externo, y entonces lo que recibirá el store para crear el médico será un arreglo de especialidades. Desde el modelo de usuarios tenemos specialties y también tenemos el método 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-
4  control selectpicker" data-style="btn-default" multiple title="
5  Seleccione una o varias">
6  @foreach($specialties as $specialty)
7  <option value="{{ $specialty->id }}">{{ $specialty->name }}</
8  option>
9  @endforeach
10 </select>
11 </div>
12
13 /***** METODO PARA CREAR MEDICO*****/
14 public function store(Request $request)
15 {
16     $rules = [
17         'name' => 'required|min:3',
18         'email' => 'required|email',
19         'dni' => 'nullable|digits:8',
20         'address' => 'nullable|min:5',
21         'phone' => 'nullable|min:6'
22     ];
23     $this->validate($request,$rules);
24
25     $user = User::create(
26         $request->only('name','email','dni','address','phone')
27         + [
28             'role' => 'doctor',
29             'password' => bcrypt($request->input('password'))
30         ]
31     );
32     // asignar especialidades al medico
33     $user->specialties()->attach($request->input('specialties'));
34     // va a recibir el arreglo de las especialidades
35
36     $notificacion = 'El MEDICO se ha registrado correctamente.';
37     return redirect('/doctors')->with(compact('notificacion'));
38 }

```

```

35 }
36

```

Listing 27: Asignar especialidades a un medico

Para listar los médicos según la especialidad tenemos varias opciones tales como usar (vue, jquery, javascript), en este caso utilizaremos jQuery. cuando laravel devuelve una colección la devuelve automáticamente 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 según la especialidad

Cada que exista un cambio en el campo specialty mandaremos llamar la ruta para listar los médicos según 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="">Selecione 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
47     }</option>';
48     });
49     $doctor.html(htmlOptions);
50     loadHours();
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,
8  ScheduleServiceInterface $scheduleService){
9      $rules = [
10         'date' => 'required|date_format:"Y-m-d"',
11         'doctor_id' => 'required|exists:users,id'
12     ];
13     $this->validate($request, $rules);
14
15     $date = $request->input('date');
16     $doctorId = $request->input('doctor_id');
17
18     return $scheduleService->getAvailableIntervals($date,
19     $doctorId);
20 }
21
22
23 /***** INTERFACE *****/
24 interface ScheduleServiceInterface
25 {
26     public function isAvailableInterval($date, $doctorId, Carbon
27     $start);
28     public function getAvailableIntervals($date, $doctorId);
29 }

```

```

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

```

```

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=${
5      {doctorId}';
6      $.getJSON(url, displayHours);
7  }
8
9  function displayHours(data){
10     if(!data.morning && !data.afternoon){
11         console.log("No se encontraron horas para el medico ese dia
12         ");
13         $hours.html(noHoursAlert);
14         return;
15     }
16 }
17

```

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,
13          $start);
14
15          $start->addMinutes(30);
16          $interval['end'] = $start->format('g:i A');
17
18          if($available){
19              $intervals [] = $interval;
20          }
21      }
22      return $intervals;
23  }
24

```

```
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,
40             $doctorId);
41         $afternoonIntervals = $this->getIntervals(
42             $workDay->afternoon_start, $workDay->afternoon_end
43             , $date, $doctorId
44         );
45         $data = [];
46         $data['morning'] = $morningIntervals;
47         $data['afternoon'] = $afternoonIntervals;
48         return $data;
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 Logging

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 contenido y match-parent es que utilize el ancho 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 medida que se usa en android no son px sino dp, pero para los textos se usa sp, puedo repartir pesos con weight.

```
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  android:padding="8dp"
9  android:background="@drawable/backgroundsss"
10 tools:context=".MainActivity">
11
12  <TextView
13  android:layout_width="match_parent"
14  android:layout_height="0dp"
15  android:layout_weight="1"
16  android:textSize="30sp"
17  android:background="@color/transparent"
18  android:gravity="center_horizontal"
19  android:text="@string/label_login"
20  android:textColor="@color/letras"
21  android:textAppearance="@style/Base.TextAppearance.AppCompat.
   Medium" />
22
23  <LinearLayout
24  android:layout_width="match_parent"
25  android:layout_height="wrap_content"
26  android:background="@color/transparent"
27  android:padding="8dp"
28  android:orientation="vertical">
29
30  <TextView
31  android:layout_width="match_parent"
32  android:layout_height="wrap_content"
33  android:gravity="center_horizontal"
34  android:layout_marginTop="12dp"
35  android:textSize="20sp"
36  android:textColor="@color/letras"
37  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
3         /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         tools:context=".MainActivity">
```

```
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
```

```

71     android:inputType="textPassword"
72     android:layout_width="match_parent"
73     android:layout_height="wrap_content"
74     android:layout_marginTop="12dp"
75     android:textColor="@color/white"/>
76
77
78     <Button
79     android:layout_width="match_parent"
80     android:layout_height="wrap_content"
81     android:layout_marginTop="12dp"
82     android:text="@string/buton_register"
83     style="@style/Base.Widget.AppCompat.Button.Colored"/>
84
85     <TextView
86     android:layout_width="match_parent"
87     android:layout_height="wrap_content"
88     android:layout_marginTop="12dp"
89     android:layout_marginBottom="16dp"
90     android:gravity="center_horizontal"
91     android:textColor="@color/white"
92     android:text="@string/go_to_login" />
93
94 </LinearLayout>
95
96
97 </LinearLayout>
98

```

Listing 34: Pantalla de Registro

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

```

1     import android.os.Bundle
2     import android.view.View
3     import android.widget.TextView
4     import android.widget.Toast
5     import kotlinx.android.synthetic.main.activity_main.*
6
7     class MainActivity : AppCompatActivity() {
8
9         override fun onCreate(savedInstanceState: Bundle?) {
10             super.onCreate(savedInstanceState)
11             setContentView(R.layout.activity_main)
12
13             tvGoToRegister.setOnClickListener {
14                 Toast.makeText(this, "Hola soy un Toast", Toast.
15                     LENGTH_SHORT)
16             }
17
18         }
19     }
20

```

Listing 35: Toast con Kotlin

Para abrir otro activity desde uno se usa el metodo Intent y startActivity

```

1     import android.os.Bundle
2     import android.view.View
3     import android.widget.TextView

```

```

4 import android.widget.Toast
5 import kotlinx.android.synthetic.main.activity_main.*
6
7 class MainActivity : AppCompatActivity() {
8
9     override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         setContentView(R.layout.activity_main)
12
13         tvGoToRegister.setOnClickListener {
14             Toast.makeText(this, getString(R.string.
15                 please_fill_your_data), Toast.LENGTH_SHORT)
16
17             // si usamos var la variable puede cambiar pero si usamos
18             // val no cambiara
19             // para no tener error tenemos que poner .java al final
20             // porque intent es una clase java
21             val intent = Intent(this, RegisterActivity::class.java)
22             startActivity(intent)
23         }
24     }
25 }
26
27
28

```

Listing 36: Intent entre pantallas

9.3. Diseño del Menú de opciones

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

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res
3 /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:gravity="center"
10    android:padding="16dp"
11    android:background="@drawable/bg_gradient"
12    tools:context=".MenuActivity">
13
14    <TextView
15        android:paddingTop="12dp"
16        android:layout_width="match_parent"
17        android:layout_height="60dp"
18        android:textSize="30sp"
19        android:gravity="center_horizontal"
20        android:text="@string/option_menu"
21        android:textColor="@color/white"
22        android:textAppearance="@style/Base.TextAppearance.AppCompat.
23        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 <RadioButton
212     android:layout_width="wrap_content"
213     android:layout_height="wrap_content"
214     android:text="8:00 AM"
215     android:textColor="@color/letras" />
216
217 <RadioButton
218     android:layout_width="wrap_content"
219     android:layout_height="wrap_content"
220     android:text="8:30 AM"
221     android:textColor="@color/letras" />
222
223 <RadioButton
224     android:layout_width="wrap_content"
225     android:layout_height="wrap_content"
226     android:text="9:00 AM"
227     android:textColor="@color/letras" />
228
229 <RadioButton
230     android:layout_width="wrap_content"
231     android:layout_height="wrap_content"
232     android:text="8:00 AM"
233     android:textColor="@color/letras" />
234
235 <RadioButton
236     android:layout_width="wrap_content"
237     android:layout_height="wrap_content"
238     android:text="8:30 AM"
239     android:textColor="@color/letras" />
240
241 <RadioButton
242     android:layout_width="wrap_content"
243     android:layout_height="wrap_content"
244     android:text="9:00 AM"
245     android:textColor="@color/letras" />
246
247 <RadioButton
248     android:layout_width="wrap_content"
249     android:layout_height="wrap_content"
250     android:text="8:00 AM"
251     android:textColor="@color/letras" />
252
253 <RadioButton
254     android:layout_width="wrap_content"
255     android:layout_height="wrap_content"
256     android:text="8:30 AM"
257     android:textColor="@color/letras" />
258
259 <RadioButton
260     android:layout_width="wrap_content"
261     android:layout_height="wrap_content"
262     android:text="9:00 AM"
263     android:textColor="@color/letras" />
264
265
266
267
268 </RadioGroup>
269
270
271 <Button
272     android:id="@+id/btnConfirmAppointment "
```

```

273 style="@style/menuButton"
274 android:layout_width="match_parent"
275 android:layout_marginTop="12dp"
276 android:layout_marginBottom="16dp"
277 android:drawableEnd="@drawable/ic_check"
278 android:text="@string/btn_confirm" />
279
280 </LinearLayout>
281 </android.support.v4.widget.NestedScrollView>
282
283 </android.support.v7.widget.CardView>
284
285
286 </LinearLayout>
287
288
289
290 /***** CODIGO PARA OCULTAR CARDVIEWS *****/
291
292 btnNext.setOnClickListener {
293     cvStep1.visibility = View.GONE
294     cvStep2.visibility = View.VISIBLE
295 }
296

```

Listing 38: Diseño primer paso reservar cita

Para mayor funcionamiento poblaremos nuestro spinner con datos ficticios.

```

1 // el material spinner arrayAdapter nos pide 3 parametros(
2 contexto, de que forma se mostraran los datos
3 // y finalmente el arreglo de datos)
4 val specialtiesOptions = arrayOf("Specialty A","Specialty B","
5 Specialty C")
6 val adapter = ArrayAdapter<String>(this, android.R.layout.
7 simple_list_item_1,specialtiesOptions)
8 spinnerSpecialties.setAdapter(adapter)
9
10
11 val doctorOptions = arrayOf("Medico 1","Medico 2","Medico 3")
12 spinnerDoctors.adapter = ArrayAdapter<String>(this,android.R.
13 layout.simple_list_item_1,doctorOptions)

```

Listing 39: Poblar spinner con datos de prueba

9.5. DatePicker Dialog

```

1 /***** XML *****/
2 <EditText
3 android:id="@+id/etScheduledDate"
4 android:onClick="onClickScheduledDate"
5 android:inputType="date"
6 android:focusable="false"
7 android:clickable="true"
8 android:maxLines="1"
9 android:layout_width="match_parent"
10 android:layout_height="wrap_content"
11 android:textColor="@color/letras" />
12
13 /***** 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 {
27                 datePicker, y, m, d ->
28                 //Toast.makeText(this,"$y-$m-$d", Toast.LENGTH_SHORT).show
29                 ()
30                 calendar.set(y,m,d)
31                 etScheduledDate.setText(resources.getString(R.string.
32                 date_format, y, m, d))
33             }
34             // creo un dialogoPicker
35             DatePickerDialog(this, listener, year, month, dayOfMonth).
36             show()
37         }
38     }

```

Listing 40: DatePicker XML y DateDialog

9.6. Generar radioButton

```

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
3     <LinearLayout
4     android:layout_width="match_parent"
5     android:layout_height="wrap_content"
6     android:orientation="horizontal">

```

```
7 <LinearLayout
8   android:id="@+id/radioGroupLeft"
9   android:layout_width="0dp"
10  android:layout_weight="1"
11  android:layout_height="wrap_content"
12  android:layout_marginBottom="8dp"
13  android:orientation="vertical">
14
15 <RadioButton
16   android:layout_width="wrap_content"
17   android:layout_height="wrap_content"
18   android:text="8:00 AM"
19   android:textColor="@color/letras" />
20
21
22 <RadioButton
23   android:layout_width="wrap_content"
24   android:layout_height="wrap_content"
25   android:text="8:00 AM"
26   android:textColor="@color/letras" />
27
28
29 </LinearLayout>
30
31
32 <LinearLayout
33   android:id="@+id/radioGroupRight"
34   android:layout_width="0dp"
35   android:layout_weight="1"
36   android:layout_height="wrap_content"
37   android:layout_marginBottom="8dp"
38   android:orientation="vertical">
39
40 <RadioButton
41   android:layout_width="wrap_content"
42   android:layout_height="wrap_content"
43   android:text="8:00 AM"
44   android:textColor="@color/letras" />
45
46
47 <RadioButton
48   android:layout_width="wrap_content"
49   android:layout_height="wrap_content"
50   android:text="8:00 AM"
51   android:textColor="@color/letras" />
52
53
54 </LinearLayout>
55
56 </LinearLayout>
57
58
59 /***** KOTLIN *****/
60 private fun displayRadioButtons(){
61     // limpiar los radioButton ya existentes
62     //radioGroup.clearCheck()
63     selectedRadioButton = null
64     // limpiar los elementos ya puestos
65     radioGroupLeft.removeAllViews()
66     radioGroupRight.removeAllViews()
67
68     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                 radioButtonGroupLeft.addView(radioButton)
84             else
85                 radioButtonGroupRight.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 citas 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  *****/
3  <?xml version="1.0" encoding="utf-8"?>
4  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
5      xmlns:app="http://schemas.android.com/apk/res-auto"
6      xmlns:tools="http://schemas.android.com/tools"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent"
9      android:orientation="vertical"
10     android:background="@drawable/bg_gradient"
11     tools:context=".AppointmentsActivity">
12
13     <TextView
14         android:layout_marginTop="12dp"
15         android:layout_width="match_parent"
16         android:layout_height="wrap_content"
17         android:textColor="@color/white"
18         style="@style/Base.TextAppearance.AppCompat.Medium"
19         android:gravity="center_horizontal"
20         android:text="@string/my_appointments"/>
21
22     <android.support.v7.widget.RecyclerView
23         android:id="@+id/rvAppointments"
24         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)
111            // va ser lineal // gridLayoutManager
112
113            val appointments = ArrayList<Appointment>()
114            appointments.add(Appointment(1,"Medico Test
115            ", "12/12/2021", "4:00 PM"))
116            appointments.add(Appointment(2,"Medico Test
117            ", "12/12/2021", "4:00 PM"))
118            appointments.add(Appointment(3,"Medico Test
119            ", "12/12/2021", "4:00 PM"))
120
121            rvAppointments.adapter = AppointmentAdapter(appointments)
122        }
123    }
124
125
126     /***** ADAPTADOR APPOINTMENTS *****/
127     package com.lopez.samuel.myappointment
128
129     import android.support.v7.widget.RecyclerView
130     import android.view.LayoutInflater
131     import android.view.View
132     import android.view.ViewGroup
133     import com.lopez.samuel.myappointment.model.Appointment
134     import kotlinx.android.synthetic.main.item_appointment.view.*
135
136     class AppointmentAdapter(private val appointments: ArrayList<
137     Appointment>): RecyclerView.Adapter<AppointmentAdapter.
138     ViewHolder>(){
139
140         // representa nuestra vista
141         class ViewHolder(itemView: View): RecyclerView.ViewHolder(
142         itemView) {
```

```

138         fun bind(appointment: Appointment) =
139             // estamos accediendo solo al objeto itemView
140             with(itemView){
141                 tvAppointmentId.text = context.getString(R.string.
142 item_appointment_id,appointment.id)
143                 tvDoctorName.text = appointment.doctorName
144                 tvScheduledDate.text = context.getString(R.string.
145 item_appointment_date,appointment.scheduledDate)
146                 tvScheduledTime.text = context.getString(R.string.
147 item_appointment_time,appointment.scheduledTime)
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.
154 item_appointment, parent, false)
155             )
156
157         // Number of Elements
158         override fun getItemCount() = appointments.size
159
160         // Bind to fate with data
161         override fun onBindViewHolder(holder: ViewHolder, position:
162 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 aspectos 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

```

1      <?xml version="1.0" encoding="utf-8"?>
2      <manifest xmlns:android="http://schemas.android.com/apk/res/
  android"
3      package="com.lopez.samuel.myappointment">
4
5      <application
6      android:allowBackup="true"
7      android:icon="@mipmap/ic_launcher"
8      android:label="@string/app_name"
9      android:roundIcon="@mipmap/ic_launcher_round"
10     android:supportsRtl="true"
11     android:theme="@style/AppTheme">
12     <activity android:name=".AppointmentsActivity"
13     android:parentActivityName=".MenuActivity"></activity>
14
15     <activity android:name=".CreateAppointmentActivity"
16     android:parentActivityName=".MenuActivity"/>
17
18     <activity android:name=".MenuActivity" />
19     <activity android:name=".RegisterActivity" />
20     <activity android:name=".MainActivity">
21     <intent-filter>
22     <action android:name="android.intent.action.MAIN" />
23
24     <category android:name="android.intent.category.LAUNCHER" />
25     </intent-filter>
26     </activity>
27     </application>
28
29     </manifest>
30

```

Listing 44: Back boton mediante parent

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

Para cerrar la actividad 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 SharedPreferences

```

1      val preferences = getSharedPreferences("general", Context.
  MODE_PRIVATE)
2      val session = preferences.getBoolean("session",false)
3
4      if(session)
5      goToMenuActivity()
6
7      btnLogin.setOnClickListener {
8      // validar en servidor
9      createSessionPreference()
10     goToMenuActivity()
11     }
12
13

```

```

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 Preference login y logout

11.4. Mejora del SharedPreferences

Se utilizara un object en vez de un class está 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     ?)
28     = when (value) {
29         is String? -> edit { it.putString(key, value) }
30         is Int -> edit { it.putInt(key, value) }
31         is Boolean -> edit { it.putBoolean(key, value) }
32         is Float -> edit { it.putFloat(key, value) }
33         is Long -> edit { it.putLong(key, value) }
34         else -> throw UnsupportedOperationException("Not yet
35         implemented")
36     }
37
38     /**
39     * finds a preference based on the given [key].
40     * [T] is the type of value
41     * @param defaultValue optional defaultValue - will take a
42     default defaultValue if it is not specified
43     */
44     inline operator fun <reified T : Any> SharedPreferences.get
45     (key: String, defaultValue: T? = null): T
46     = when (T::class) {
47         String::class -> getString(key, defaultValue as? String
48         ?: "") as T
49         Int::class -> getInt(key, defaultValue as? Int ?: -1) as
50         T
51         Boolean::class -> getBoolean(key, defaultValue as?
52         Boolean ?: false) as T
53         Float::class -> getFloat(key, defaultValue as? Float ?
54         : -1f) as T
55         Long::class -> getLong(key, defaultValue as? Long ?
56         : -1) as T
57         else -> throw UnsupportedOperationException("Not yet
58         implemented")
59     }
60 }
61
62 /***** CLASE MAIN *****/
63 val preferences = PreferenceHelper.defaultPrefs(this) // va ser
64 por default
65
66 if(preferences["session",false])
67     goToMenuActivity()
68
69 private fun createSessionPreference(){
70     /**
71     val preferences = getSharedPreferences("general", Context.
72     MODE_PRIVATE)
73     val editor = preferences.edit()
74     editor.putBoolean("session",true)
75     editor.apply()
76     */
77     val preferences = PreferenceHelper.defaultPrefs(this)
78     preferences["session"] = true

```

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

Listing 46: Mejora

11.5. Snackbar by Lazy

Para esto debemos sobrescribir el metodo `onBackPressed()` 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 metodo `onBackPressed` y ver que el snackbar ya esta visible entonces se sale de la app.

```
1     private val snackbar = Snackbar(vista, texto, duracion)
2
3     override 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 cualquiera 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 identificar si ha sido alterado y que el destinatario sepa si fue modificado, la entrada es la misma siempre va a ser el mismo resultado, no

es reversible. También podemos ofuscar el código que significa que las variables no describen lo que hacen y no tienen saltos de línea.

Hay muchas maneras de autenticar, una manera es la autenticación basada en tokens, en el servidor no se guardan ningún token simplemente se asegura que sea válido el token, simplemente se firma y a través de esta firma el servidor verifica si es válido y fue emitido por 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 Authorization Header,

El token está compuesto de un Header, un payload y una firma, paso 1.- header.payload.firma, el header contiene el tipo y el algoritmo, 2.-payload contiene claims, son datos que contienen información, iss nombre o dominio, sub descripción exp cuando va a expirar el token, 3.- crear la firma, se codifica el header y el payload con un algoritmo base64 y se concatenan, luego a ese resultado se aplica un algoritmo de hashing y con un secret, después se vuelve a hashear 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: NlhVFJ418CP7ynImVfPK64RhgdhSqk7

12. Retrofit con Kotlin

Para instalar Kotlin fue necesario copiar y pegar en el build.gradle la siguiente dependencia es implementation 'com.squareup.retrofit2:retrofit:2.3.0' pero también tiene otra dependencia que ayuda a convertir los objetos Json que vienen del servidor a Objetos Kotlin automáticamente esa dependencia es implementation 'com.squareup.retrofit2:converter-gson:2.3.0', también 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 un 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 utilice 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:
11                 Throwable) {
12                 Toast.makeText(this@CreateAppointmentActivity, getString(R
13                     .string.error_loading_speialties), Toast.LENGTH_SHORT).show()
14                 finish()
15             }
16
17             override fun onResponse(call: Call<ArrayList<Specialty>>,
18                 response: Response<ArrayList<Specialty>>) {
19                 if(response.isSuccessful){ // cuando es correcto esta
20                     entre 200 y 300

```

```

17         val specialties = response.body() // obtenemos un
           arrayList de especialidades
18
19         val specialtyOptions = ArrayList<String>()
20         specialties?.forEach{
21             specialtyOptions.add(it.name)
22         }
23         spinnerSpecialties.adapter = ArrayAdapter<String>(
           this@CreateAppointmentActivity, android.R.layout.
           simple_list_item_1, specialtyOptions)
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 utilizaremos un `clickSelecteItem` para cuando se haga un modificacion del spinner especialidades entonces se mande a llamar la ruta y así 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     *****/
8     @GET("specialties/{specialty}/doctors")
9     abstract fun getDoctors(@Path("specialty") specialtyId: Int):
10         Call<ArrayList<Doctor>>
11
12     /***** Esperar cambios del spinner specialties
13     *****/
14     private fun listenSpecialtyCHange(){
15         // obtengo el id del spinner Specialties
16         spinnerSpecialties.onItemSelectedListener = object:
17             AdapterView.OnItemSelectedListener{
18                 override fun onNothingSelected(parent: AdapterView<*>?) {
19
20                 }
21
22                 override fun onItemSelected(adapter: AdapterView<*>?, view:
23                     View?, position: Int, id: Long) {
24                     // adapter nos permitira ingresar a los id
25                     val specialty = adapter?.getItemAtPosition(position) as
26                     Specialty
27                     Toast.makeText(this@CreateAppointmentActivity, "Id: ${
28                     specialty.id}", Toast.LENGTH_SHORT).show()
29                     loadDoctors(specialty.id)
30                 }
31             }
32     }
33 }
34

```

```
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:
36              Throwable) {
37              Toast.makeText(this@CreateAppointmentActivity,getString(R.
38                  string.error_loading_doctors), Toast.LENGTH_SHORT).show()
39
40              override fun onResponse(call: Call<ArrayList<Doctor>>,
41                  response: Response<ArrayList<Doctor>>) {
42                  if(response.isSuccessful){ // cuando es correcto esta entre
43                      200 y 300
44                      val doctors = response.body() // obtenemos un arrayList
45                      de especialidades
46
47                      spinnerDoctors.adapter = ArrayAdapter<Doctor>(
48                          this@CreateAppointmentActivity, android.R.layout.
49                          simple_list_item_1,doctors)
50                  }
51              }
52      })
53  }
```

Listing 50: Spinner de medicos