

ódigo Android7.116



DEPARTAMENTO DE TECNOLOGÍA DIGITAL
DISEÑO Y DESARROLLO DE SOFTWARE

SISTEMA DE SERVICIOS MULTIFUNCIONALES
DOMÉSTICOS

PRIMERA ENTREGA

Presentado por
Xiomara Ainhoa Torres Galindo
Lourdes Isabel Pampa Manrique
Brian Jossep Villanueva Vargas

Docente
Jaime Gómez Marin

Lima , 2019

Índice general

1. Introducción	4
1.1. Propósito	4
1.2. Alcance	4
1.3. Definiciones, siglas y abreviaturas	5
1.4. Referencias	5
2. Visión del Proyecto	6
3. Equipo	7
4. Product Backlog	8
5. Historias de Usuarios	9
6. Planificación	11
7. Organización del Proyecto	13
8. Referencia Bibliograficas	25
9. Anexos	26
9.1. GitHub	26
9.2. Trello - Sprint 1	26

Índice de figuras

7.1.	Código Android	16
7.2.	Interfaz Android	17
7.3.	Spring Boot	18
7.4.	Spring Boot	19
7.5.	Spring Boot	20
7.6.	React	21
7.7.	React	22
7.8.	React	23
7.9.	React	24

Índice de cuadros

Capítulo 1

Introducción

1.1. Propósito

- Nuestro sistema se basa en crear un sistema y una aplicación móvil dedicada a ofrecer servicios multifuncionales (arreglos, reparaciones e instalaciones) para los hogares en varios sectores de Lima Metropolitana. Para esto contaremos con personas especializadas en 3 rubros (carpintería, gasfitería y electricista), se podrá hacer una clasificación de nuestros servicios en función a la frecuencia con la que se realicen.

1.2. Alcance

- Tal y como se ha indicado, el sistema de servicios domésticos va a ofrecer diversas actividades relacionadas con el mantenimiento y reparaciones dentro de las viviendas, donde se pueden contemplar diversos rubros de servicios como son el de electricista, carpintería y gasfitería. Entre los distintos tipos de instalaciones se encuentran: -Instalaciones eléctricas, calefacción, aire acondicionado, etc. -Instalación de puertas, muebles. -Reparación y mantenimiento de cocinas y electrodomésticos.

1.3. Definiciones, siglas y abreviaturas

- Django: Es el framework que nos permitirá diseñar el sistema en la página web , realizar las aplicaciones de cualquier complejidad a corto plazo.
- Springboot Spring boot nos permitirá centrar y simplificar el proceso de nuestro sistema.
- Android Android es el sistema operativo gratuito y con una plataforma muy amplia , lo usaremos para la implementación en el móvil.

1.4. Referencias

- <https://openwebinars.net/blog/que-es-django-y-por-que-usarlo/>
- <https://www.xatakandroid.com/sistema-operativo/que-es-android>
- <https://www.arquitecturajava.com/que-es-spring-boot/>

Capítulo 2

Visión del Proyecto

- Desarrollar un aplicativo móvil que facilite la búsqueda de los trabajadores ya sean estos electricistas, carpinteros y gasfiteros, dando la elección de los más cercanos a su ubicación.

Capítulo 3

Equipo

- El equipo scrum trabaja con cada tarea que es asignada dependiendo del Sprint y de sus roles, Xiomara Torres trabaja la parte del Front-end en la página web del proyecto, Lourdes Pampa trabaja la parte del Back-end en la página web del proyecto y Brian Villanueva trabaja en la aplicación móvil del proyecto.

Equipo	Nombre	Rol
Equipo Scrum	Xiomara Torres	Scrum Master
	Lourdes Pampa	Desarrollador
	Brian Villanueva	Desarrollador

Capítulo 4

Product Backlog

Item	Detalle	Prioridad	Dias(Estimado)
PB-01	Definir las especialidades de los servicios domésticos	60	3
PB-02	Permitir que los servicios esten al alcance de los clientes los 7 días de la semana	70	5
PB-03	Sistema de registro e inicio de sesión	30	4
PB-04	Sistema de registro de empleados	80	4
PB-05	Sistema de registro de servicios domésticos	50	4
PB-06	Permitir que el cliente tenga la información personal del trabajador	80	5
PB-07	Permitir que los trabajadores tengan más de una especialidad	40	8

Capítulo 5

Historias de Usuarios

PB	Item	Requerimiento	Beneficio	Priorid	Días
PB-01	HU-01	Yo como usuario deberia contar con las especialidades más demandadas de Lima Metropolitana	A fin de no limitar mis requerimientos.	70	5
PB-02	HU-02	Yo como usuario deberia obtener los servicios domésticos los 7 días de la semana	A fin de que mis emergencias no influyan en la disponibilidad de los trabajadores.	50	3
PB-03	HU-03	Yo como usuario deberia registrarme e iniciar sesión	A fin de poderme identificar.	80	4
PB-04	HU-04	Yo como usuario deberia visualizar el perfil de todos los trabajadores	A fin de que se puedan identificar.	60	7
PB-05	HU-05	Yo como usuario deberia visualizar los servicios domésticos ya cumplidos	A fin de que los trabajadores puedan registrar sus servicios realizados.	40	5

Capítulo 6

Planificación

SPRINT - 1

HU-03	Encar. Web	Encar. Movil	Horas
Realizar un prototipo de registro	Xiomara Torres	Brian Villanueva	2
Realizar un prototipo de inicio de sesión	Xiomara Torres	Brian Villanueva	2
Realizar una base de datos modelo entidad relación	Lourdes	Pampa	9
Realizar una sistema de registro e inicio de sesión front end	Xiomara Torres	Brian Villanueva	6
Realizar una sistema de registro e inicio de sesión back end	Lourdes Pampa	Brian Villanueva	7
Realizar tabla de clientes	Lourdes	Pampa	5

SPRINT - 2

HU-03	Encar. Web	Encar. Movil	Horas
Realizar un prototipo del listado de las especialidades	Xiomara Torres	Brian Villanueva	6
Realizar la interfaz del listado de las especialidades front-end	Xiomara Torres	Brian Villanueva	3
Realizar el listado de las especialidades en back-end	Lourdes	Pampa	7
Realizar una investigación de las especialidades más demandadas en Lima Metropolitana	Brian	Villanueva	3

Capítulo 7

Organización del Proyecto

- Se utilizó el software Trello para llevar administrar el proyecto a través de tareas que estan agrupadas en segmentos como; To do, Doing, Test, Done, Stop. En donde aparece el nombre del responsable y el seguimiento de las horas ejecutadas.
- Se utilizó el Burn Down en Excel para poder tener un estimado de como se estuvo trabajando durante el sprint, con los promedios de las horas trabajadas entre los miembros del equipo scrum, debe coincidir con las horas trabajadas en Trello.

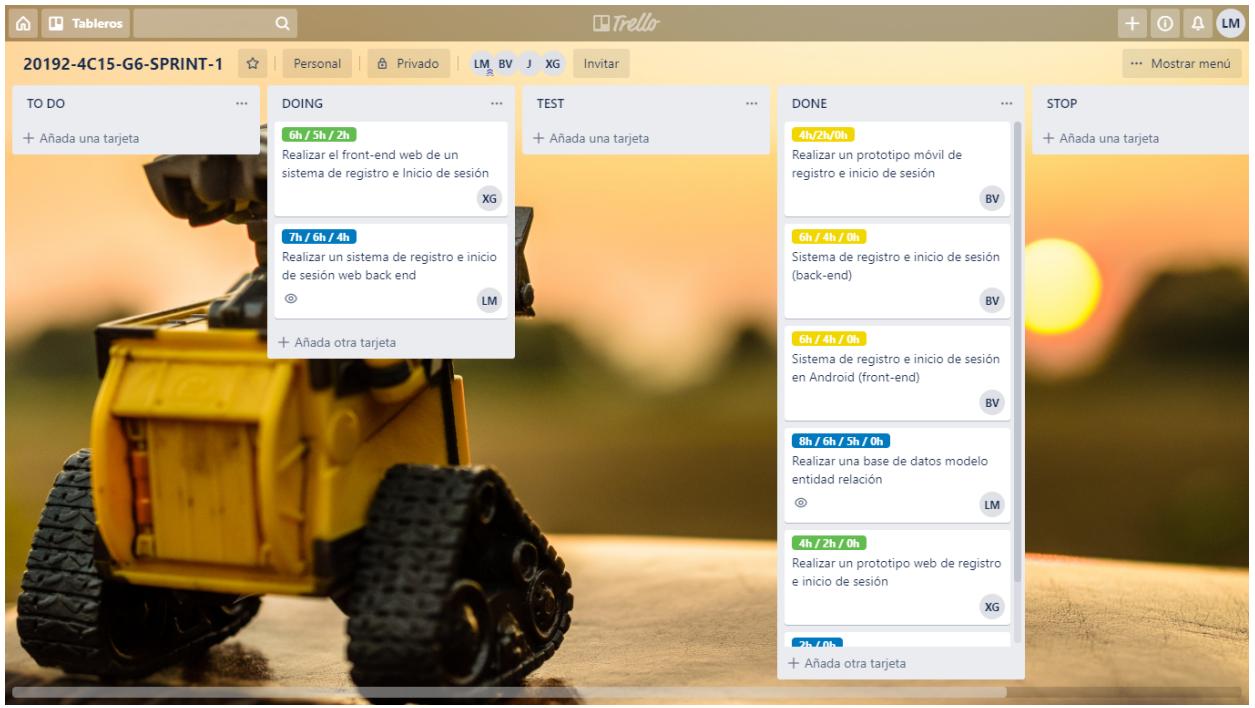
Sprint - 1

Sprint - 2

Android - Brian Villanueva

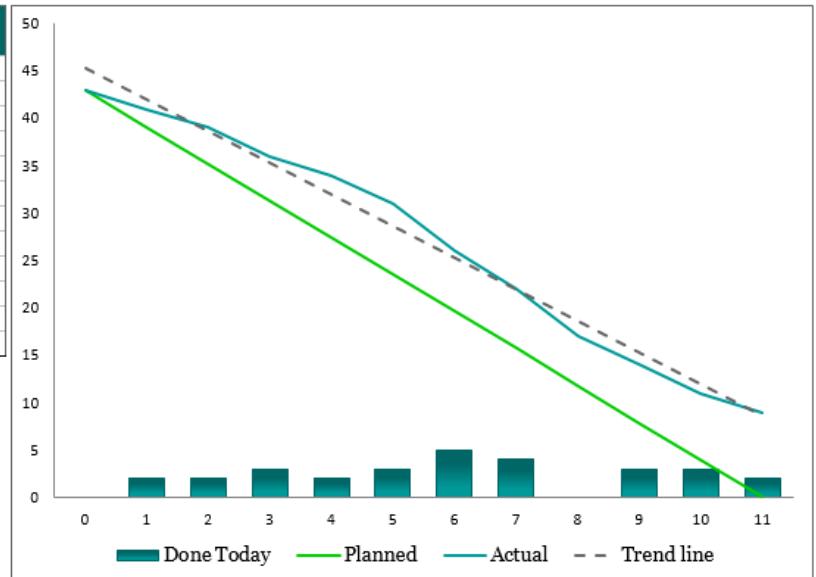
Web Back-end - Lourdes Pampa

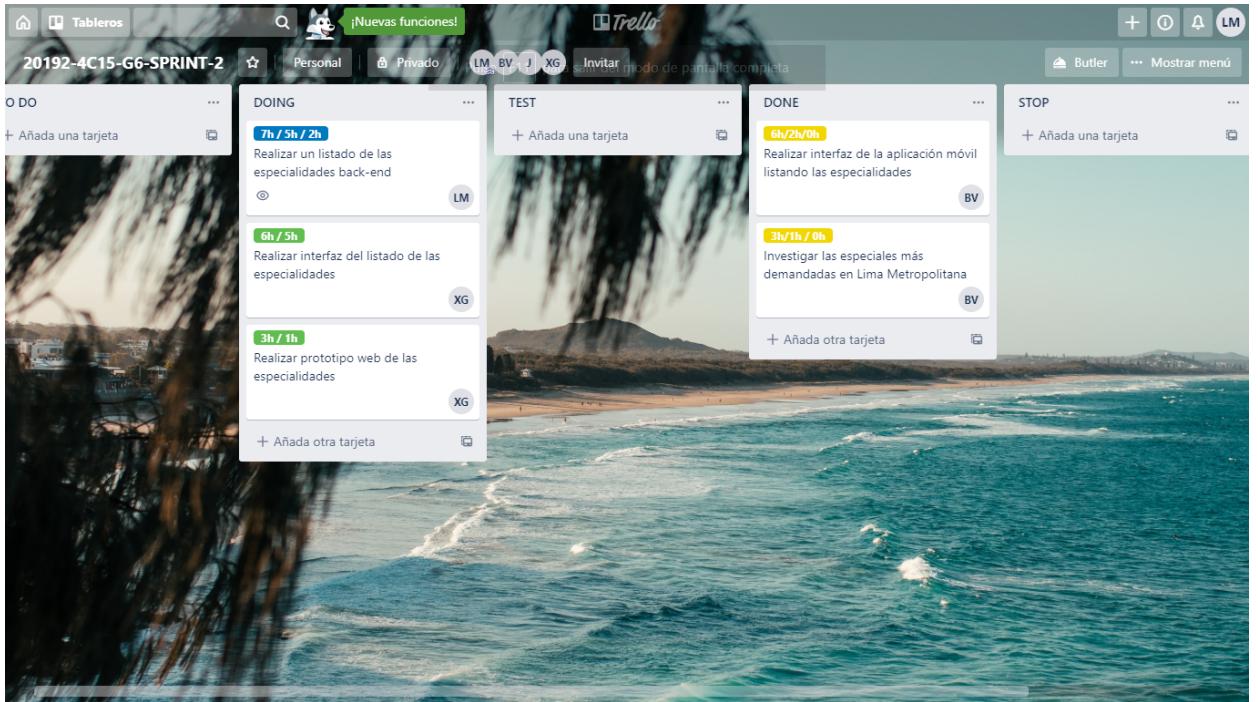
Web Front-end - Xiomara Torres



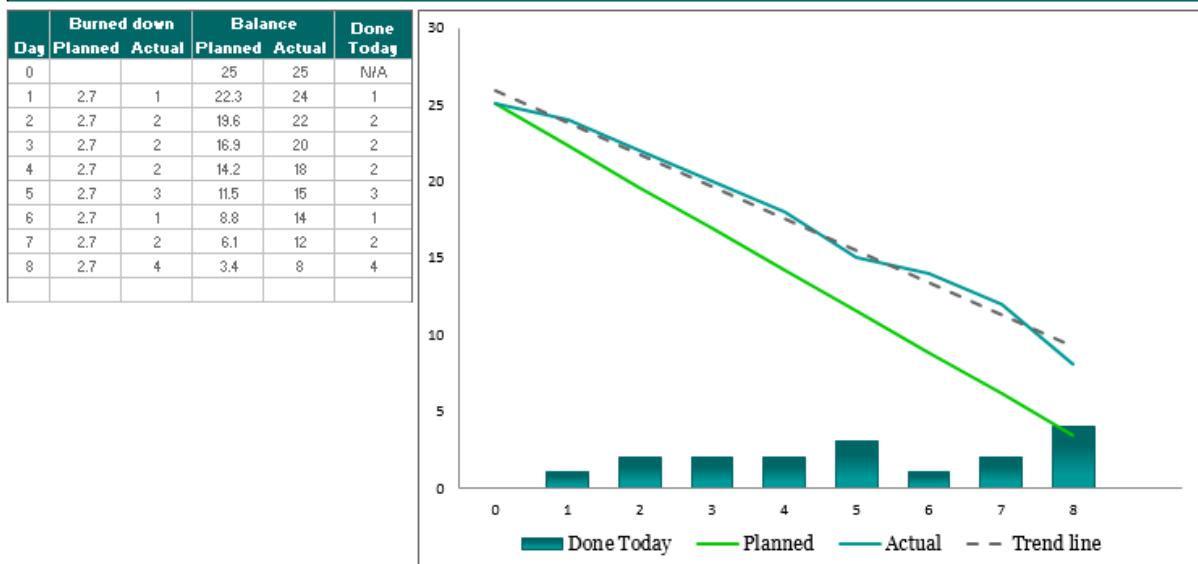
Burn Down Chart

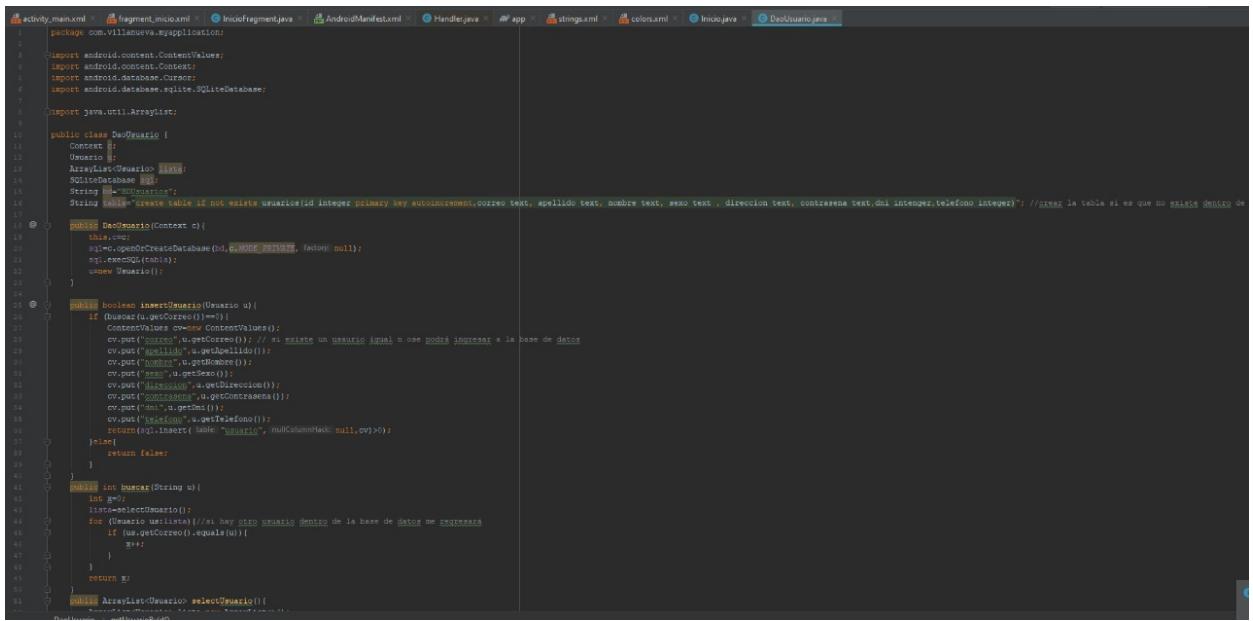
Day	Burned down		Balance		Done Today
	Planned	Actual	Planned	Actual	
0			43	43	N/A
1	3.9	2	39.1	41	2
2	3.9	2	35.2	39	2
3	3.9	3	31.3	36	3
4	3.9	2	27.4	34	2
5	3.9	3	23.5	31	3
6	3.9	5	19.6	26	5
7	3.9	4	15.7	22	4
8	3.9	5	11.8	17	0
9	3.9	3	7.9	14	3
10	3.9	3	4	11	3
11	3.9	2	0.1	9	2





Burn Down Chart





The screenshot shows the Java code for a DAO class named `DaoUsuario`. The code is used to interact with a SQLite database to manage user data. It includes methods for creating the database table if it doesn't exist, inserting a new user, and selecting users by email.

```
1 package com.villanueva.myapplication;
2
3 import android.content.ContentValues;
4 import android.content.Context;
5 import android.database.Cursor;
6 import android.database.sqlite.SQLiteDatabase;
7
8 import java.util.ArrayList;
9
10 public class DaoUsuario {
11     Context c;
12     Usuarios u;
13     ArrayList<Usuarios> lista;
14     SQLiteDatabase db;
15     String sql="CREATE TABLE IF NOT EXISTS usuarios(id integer primary key autoincrement,correo text, apellido text, nombre text, sexo text, direccion text, contrasena text, dni integer, telefono integer)"; //crea la tabla si es que no existe dentro de
16
17     public DaoUsuario(Context c){
18         this.c=c;
19         db=c.openOrCreateDatabase("bd", MODE_PRIVATE, null);
20         db.execSQL(sql);
21         db.close();
22     }
23
24     public boolean insertUsuario(Usuarios u){
25         if (u.getCorreo()!="" ){
26             ContentValues cv=new ContentValues();
27             cv.put("correo",u.getCorreo()); // si existe un usuario igual no se podrá ingresar a la base de datos
28             cv.put("apellido",u.getApellido());
29             cv.put("nombre",u.getNombre());
30             cv.put("sexo",u.getSexo());
31             cv.put("direccion",u.getDireccion());
32             cv.put("contrasena",u.getContrasena());
33             cv.put("dni",u.getDni());
34             cv.put("telefono",u.getTelefono());
35             return (db.insert("usuarios", null, cv)>0);
36         }else{
37             return false;
38         }
39     }
40
41     public int bucar(String u){
42         int g=0;
43         Usuarios seleccionarUsuario();
44         for (Usuarios usuario://si hay otro usuario dentro de la base de datos me asignará
45             if (usuario.getCorreo().equals(u)){
46                 g++;
47             }
48         }
49         return g;
50     }
51     public ArrayList<Usuarios> selectUsuario(){
52         ArrayList<Usuarios> lista = new ArrayList<Usuarios>();
53         Cursor cursor=db.rawQuery("SELECT * FROM usuarios", null);
54         while(cursor.moveToNext()){
55             Usuarios u=new Usuarios();
56             u.setId(cursor.getInt(0));
57             u.setApellido(cursor.getString(1));
58             u.setNombre(cursor.getString(2));
59             u.setSexo(cursor.getString(3));
60             u.setDireccion(cursor.getString(4));
61             u.setContrasena(cursor.getString(5));
62             u.setDni(cursor.getInt(6));
63             u.setTelefono(cursor.getInt(7));
64             u.setCorreo(cursor.getString(8));
65             lista.add(u);
66         }
67         return lista;
68     }
69 }
```

Figura 7.1: Código Android

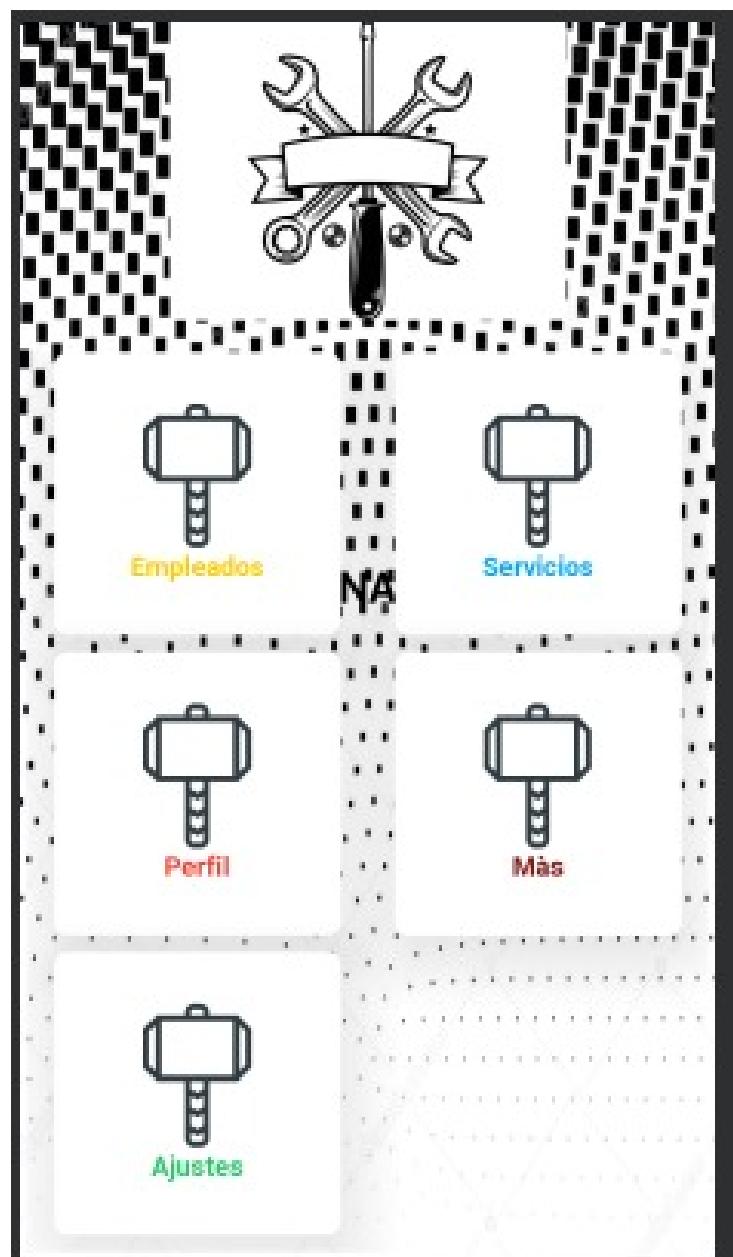
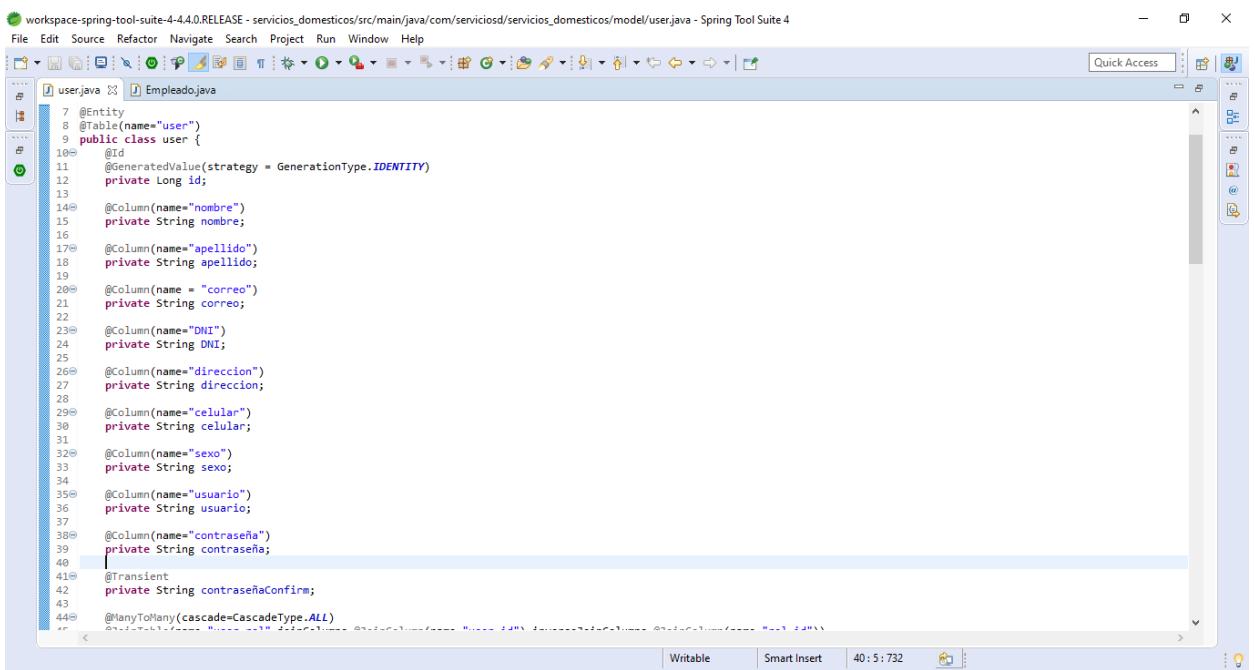


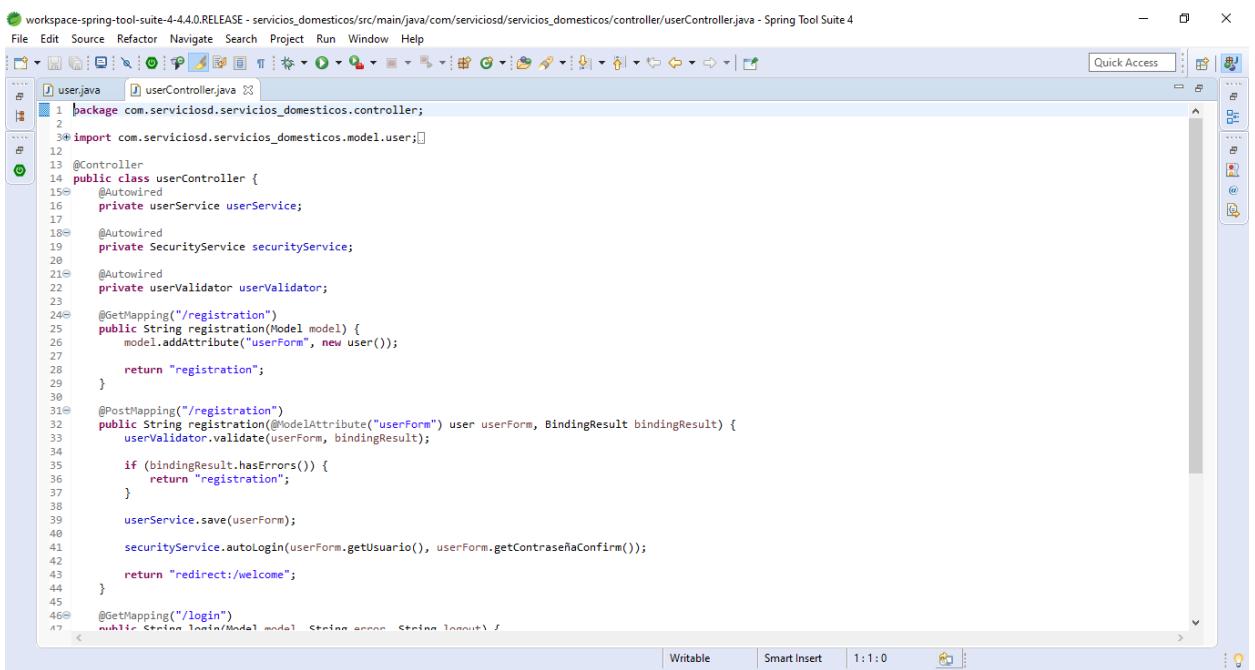
Figura 7.2: Interfaz Android



The screenshot shows the Spring Tool Suite 4 interface. The title bar indicates the workspace is named 'workspace-spring-tool-suite-4-4.4.0.RELEASE - servicios_domesticos/src/main/java/com/serviciosdomesticos/model/user.java - Spring Tool Suite 4'. The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations like Open, Save, and Build. The code editor displays two files: 'user.java' and 'Empleado.java'. The 'user.java' file is the active tab, showing Java code for an entity class named 'user' with annotations for columns and relationships. The code is as follows:

```
7 @Entity
8 @Table(name="user")
9 public class user {
10    @Id
11    @GeneratedValue(strategy = GenerationType.IDENTITY)
12    private Long id;
13
14    @Column(name="nombre")
15    private String nombre;
16
17    @Column(name="apellido")
18    private String apellido;
19
20    @Column(name = "correo")
21    private String correo;
22
23    @Column(name="DNI")
24    private String DNI;
25
26    @Column(name="direccion")
27    private String direccion;
28
29    @Column(name="celular")
30    private String celular;
31
32    @Column(name="sexo")
33    private String sexo;
34
35    @Column(name="usuario")
36    private String usuario;
37
38    @Column(name="contraseña")
39    private String contraseña;
40
41    @Transient
42    private String contraseñaConfirm;
43
44    @ManyToMany(cascade=CascadeType.ALL)
```

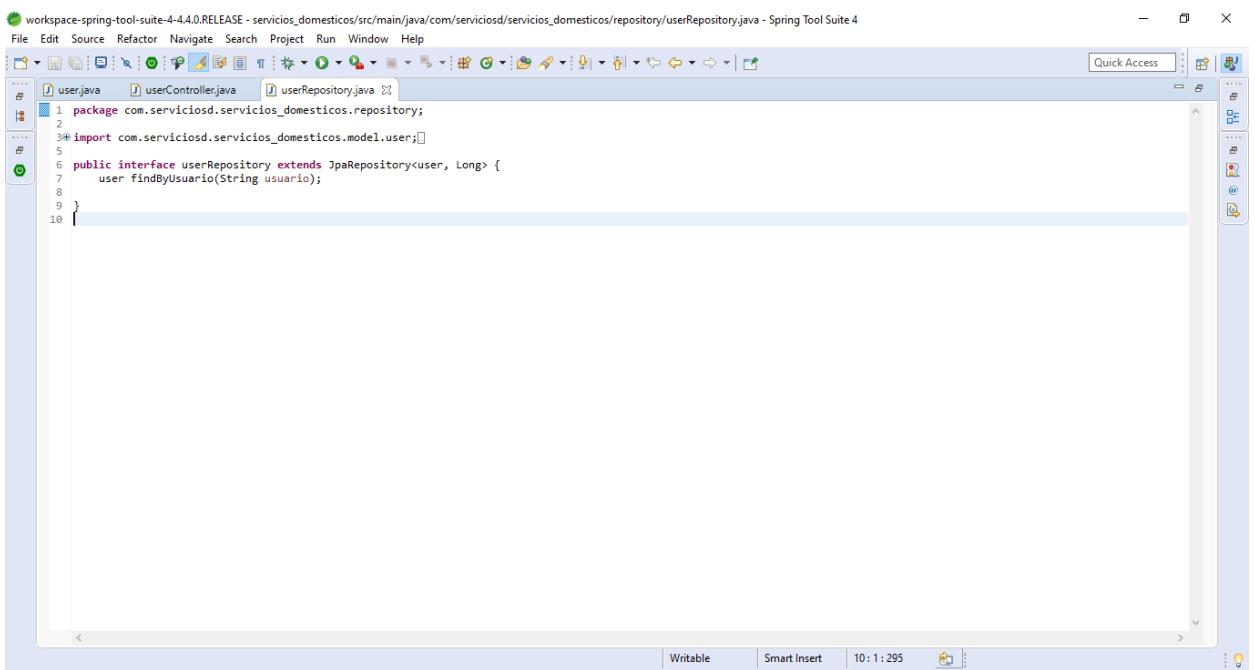
Figura 7.3: Spring Boot



The screenshot shows the Spring Tool Suite 4 interface with the file `userController.java` open in the editor. The code implements a `userController` class with methods for registration and login. The registration method handles both GET and POST requests, validating the user form and saving it to the database before redirecting to the welcome page. The login method handles a GET request for the login page.

```
1 package com.serviciosd.servicios_domesticos.controller;
2
3 import com.serviciosd.servicios_domesticos.model.user;
4
5 @Controller
6 public class userController {
7     @Autowired
8     private userService userService;
9
10    @Autowired
11    private SecurityService securityService;
12
13    @Autowired
14    private userValidator userValidator;
15
16    @GetMapping("/registration")
17    public String registration(Model model) {
18        model.addAttribute("userForm", new user());
19
20        return "registration";
21    }
22
23    @PostMapping("/registration")
24    public String registration(@ModelAttribute("userForm") user userForm, BindingResult bindingResult) {
25        userValidator.validate(userForm, bindingResult);
26
27        if (bindingResult.hasErrors()) {
28            return "registration";
29        }
30
31        userService.save(userForm);
32
33        securityService.autoLogin(userForm.getUsuario(), userForm.getContrasenaConfirm());
34
35        return "redirect:/welcome";
36    }
37
38    @GetMapping("login")
39    public String login(Model model, String error, String logout) {
40
41        return "login";
42    }
43}
```

Figura 7.4: Spring Boot



The screenshot shows the Spring Tool Suite 4 interface. The title bar reads "workspace-spring-tool-suite-4-4.4.0.RELEASE - servicios_domesticos/src/main/java/com/serviciosd/servicios_domesticos/repository/userRepository.java - Spring Tool Suite 4". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like Open, Save, and Build. The code editor window displays the following Java code:

```
1 package com.serviciosd.servicios_domesticos.repository;
2
3 import com.serviciosd.servicios_domesticos.model.user;
4
5 public interface userRepository extends JpaRepository<user, Long> {
6     user findByUsuario(String usuario);
7 }
8
9
10 }
```

Figura 7.5: Spring Boot

```

62         </div>
63     <div class="form-check">
64         <input class="form-check-input" type="radio" name="genero" value="m"/>
65         <label class="form-check-label">Masculino </label>
66     </div>
67     <div class="form-group">
68         <label>DNI </label>
69         <input type="text" name="dni" id="dni"
70             class="form-control"
71             value={this.state.dni}
72             onChange={this.onChange.bind(this)}/>
73     </div>
74 </div>
75
76 <div class="col">
77     <div class="form-group">
78         <label>Cargo </label>
79         <select class="custom-select mr-sm-2" id="cargo" name="cargo"
80             value={this.state.cargo} onChange={this.onChange.bind(this)}>
81             <option value="">Selecione</option>
82             <option value="electricista">Electricista</option>
83             <option value="carpintero">Carpintero</option>
84             <option value="gafitero">Gásfitero</option>
85         </select>
86     </div>
87     <div class="form-group">
88         <label>Email </label>
89         <input type="email" name="email" id="email"
90             class="form-control"
91             value={this.state.email}
92             onChange={this.onChange.bind(this)}/>
93     </div>

```

Figura 7.6: React

```
14  onChange(e){
15    this.setState({
16      [e.target.name]:e.target.value
17    })
18  }
19
20  render(){
21    return(
22      <div class = "lista-page" id="lista">
23        <h1>Oficios</h1>
24        <div class="container">
25          <div class="card">
26            <h3>Carpintero</h3>
27            
28          </div>
29          <div class="card">
30            <h3>Electricista</h3>
31            
32          </div>
33          <div class="card">
34            <h3>Gasfitero</h3>
35            
36          </div>
37        </div>
38      </div>
39    )
40  }
41}
42
43 export default Lista;
```

Figura 7.7: React

```
13
14  h1{
15      text-align: center;
16  }
17
18  h3{
19      text-align: center;
20  }
21
22  .profesion {
23      height:200px;
24  }
25
26  .container{
27      display: flex;
28      width: 100%;
29      max-width: 1200px;
30      align-items: center;
31      justify-content: center;
32      padding: 0 20px;
33  }
34
35  .card{
36      width: calc(33% - 20px);
37      height: 200px;
38      margin: 50px;
39      display: flex;
40      flex-direction: column;
41      justify-content: center;
42  }
```

Figura 7.8: React
23

```

34     render(){
35         return(
36             <div class="container" id="contenedor">
37                 <div>
38                     <img url="menu-bg.jpg"/>
39                 </div>
40                 <form class="register-form">
41                     <h2>Registro de Empleado</h2>
42                     <div class="row">
43                         <div class="col">
44                             <div class="form-group">
45                                 <label>Nombres </label>
46                                 <input type="text" name="name" id="name"
47                                     class="form-control"
48                                     value={this.state.name}
49                                     onChange={this.onChange.bind(this)}/>
50                             </div>
51                             <div class="form-group">
52                                 <label>Apellidos </label>
53                                 <input type="text" name="lastname" id="lastname"
54                                     class="form-control"
55                                     value={this.state.lastname}
56                                     onChange={this.onChange.bind(this)}/>
57                         </div>
58                         <label>Genero</label>
59                         <div class="form-check">
60                             <input class="form-check-input" type="radio" name="genero" value="f"/>
61                             <label class="form-check-label">Femenino </label>
62                         </div>
63                         <div class="form-check">
64                             <input class="form-check-input" type="radio" name="genero" value="m"/>

```

Figura 7.9: React

Capítulo 8

Referencia Bibliograficas

- <https://openwebinars.net/blog/que-es-django-y-por-que-usarlo/>
- <https://www.xatakandroid.com/sistema-operativo/que-es-android>
- <https://www.arquitecturajava.com/que-es-spring-boot/>
- <https://devcode.la/blog/frontend-y-backend/>
- <https://www.empresaeiniciativaemprededora.com/?Montar-una-Empresa-Multiservicios>

Capítulo 9

Anexos

9.1. GitHub

- <https://github.com/Lourdes1514/Proyecto-Integrador-6>

9.2. Trello - Sprint 1

- <https://trello.com/b/vi8UXfic/20192-4c15-g6-sprint-1>

9.3. Trello - Sprint 2

- <https://trello.com/b/xzMRe8AJ/20192-4c15-g6-sprint-2>