

COMPROBACIONES PRIMER SPRINT CON EVIDENCIAS

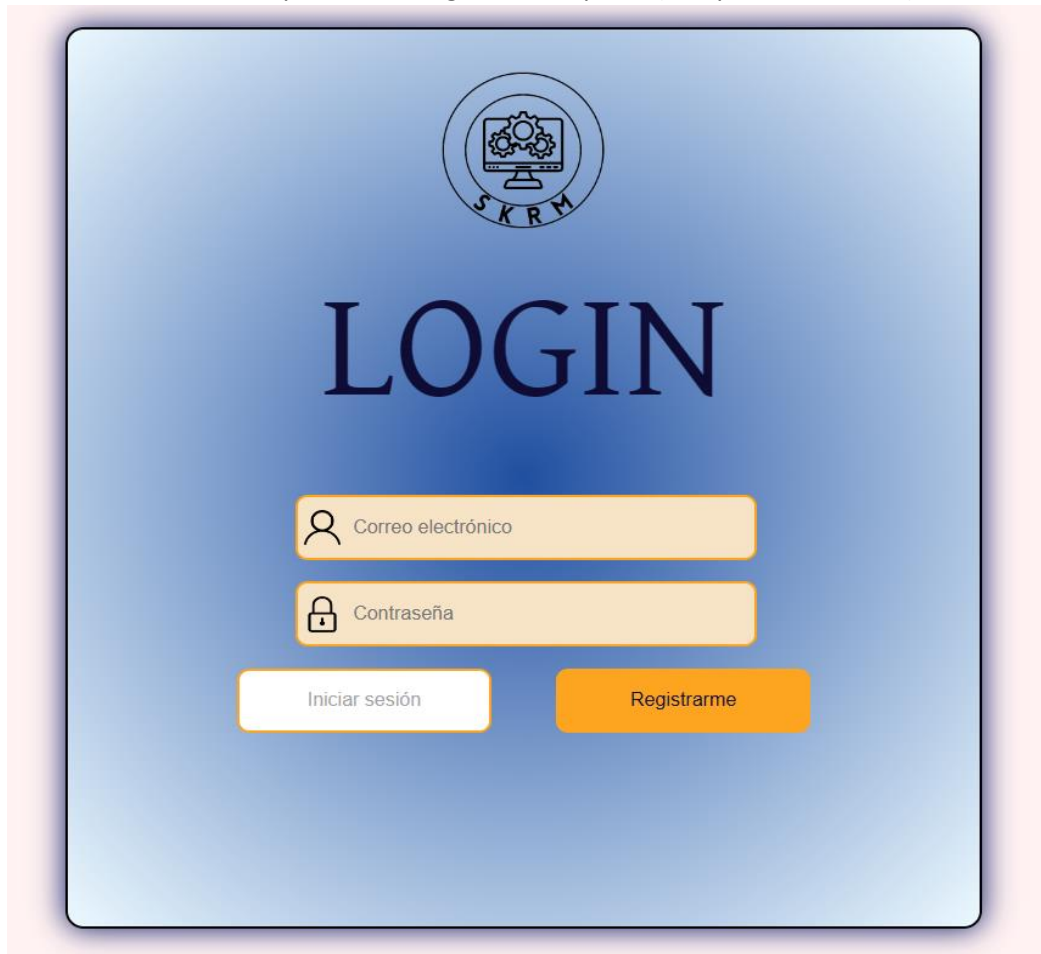


EQUIPO TESTING

ADMINISTRADOR: Los requisitos que se impusieron al principio del sprint para el equipo dedicado al administrador son los siguientes:

- Realizar la función login: El equipo ha diseñado una interfaz donde el usuario puede registrarse e iniciar sesión siguiendo la gama de colores y la mayoría de las especificaciones acordadas por todos los grupos de front, aunque faltaría el footer común de “SKRM: Todos los derechos reservados”. El diseño es bastante intuitivo y agradable para la vista cumpliendo con la regla de usabilidad aunque se deben efectuar ciertas mejoras de cara a hacerlo aún más usable (letra de información extremadamente pequeña, letra de error también muy pequeña y algo ilegible por el color rojo).

Se adjunta captura de la página principal que verifica el color azul, cajas en amarillo y bordes redondos acompañado del logo de la empresa (no aparece el footer).



Captura de la letra extremadamente pequeña de información y de error:

Regístrate

Dirección de correo electrónico inválida

 Contraseña no válida

[¿Ya tienes una cuenta? Iniciar Sesión](#)

En la parte de back, el código presenta comentarios significativos que ayudan al entendimiento y legibilidad del código, el código parece correcto pero debido a los fallos de conexión que ha habido entre todos los grupos no ha sido posible testear específicamente si han cumplido su función y el requisito de manejar las peticiones con lo obtenido de controlador. La clase User presenta los atributos de instancia en publico no cumpliendo con buenas prácticas de programación.

Captura de comentarios en el código:

```

/* Servicio CreateUser:
 * Este metodo gestiona el Register de usuarios
 * Recibe un objeto de tipo usuario y lo manda con un POST al controlador.
 * Devuelve 0 si la respuesta se ha recibido correctamente, y -1 en caso contrario
 */
public int createUser(User user) throws IOException {

    UserResponse userRes = new UserResponse(user);
    JSONObject json = new JSONObject(userRes);
    RequestBody body = RequestBody.create(json.toString(), MediaType.parse("application/json; charset=utf-8"));
    Request request = new Request.Builder().url("http://ismi.fi.upm.es:8086/crearUsuario").post(body).build();
    Call call = client.newCall(request);
    Response response = call.execute();

    if(!response.isSuccessful()) return -1;

    int res = new GsonBuilder().setPrettyPrinting().create().fromJson(response.body().string(), Integer.class);
    return res;
}

```

Captura de la clase User:

```

3 public class User {
4     public String name;
5     public String mail;
6     public String apellidos;
7
8     public String password;
9     public String dni;
10
11     public User(String mail, String password){
12         this.mail=mail;
13         this.name=name;
14     }
15
16 }

```

Captura error de Swagger:

Code	Details
500 <i>Undocumented</i>	<p>Error:</p> <p>Response body</p> <pre>{ "timestamp": "2023-02-24T19:18:21.526+00:00", "status": 500, "error": "Internal Server Error", "path": "/Login" }</pre> <p>Response headers</p> <pre>connection: close content-type: application/json date: Fri, 24 Feb 2023 19:18:21 GMT transfer-encoding: chunked vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers</pre>

CALCULADORA: El equipo de calculadora partía de los siguientes requisitos para el primer sprint:

- Diseñar métrica de costes: El equipo ha elaborado una memoria donde se detalla de manera clara y concisa cómo han llegado a obtener las fórmulas para calcular el coste total teniendo en cuenta los precios de las placas fotovoltaicas, los componentes, la mano de obra, la zona elegida y el área.

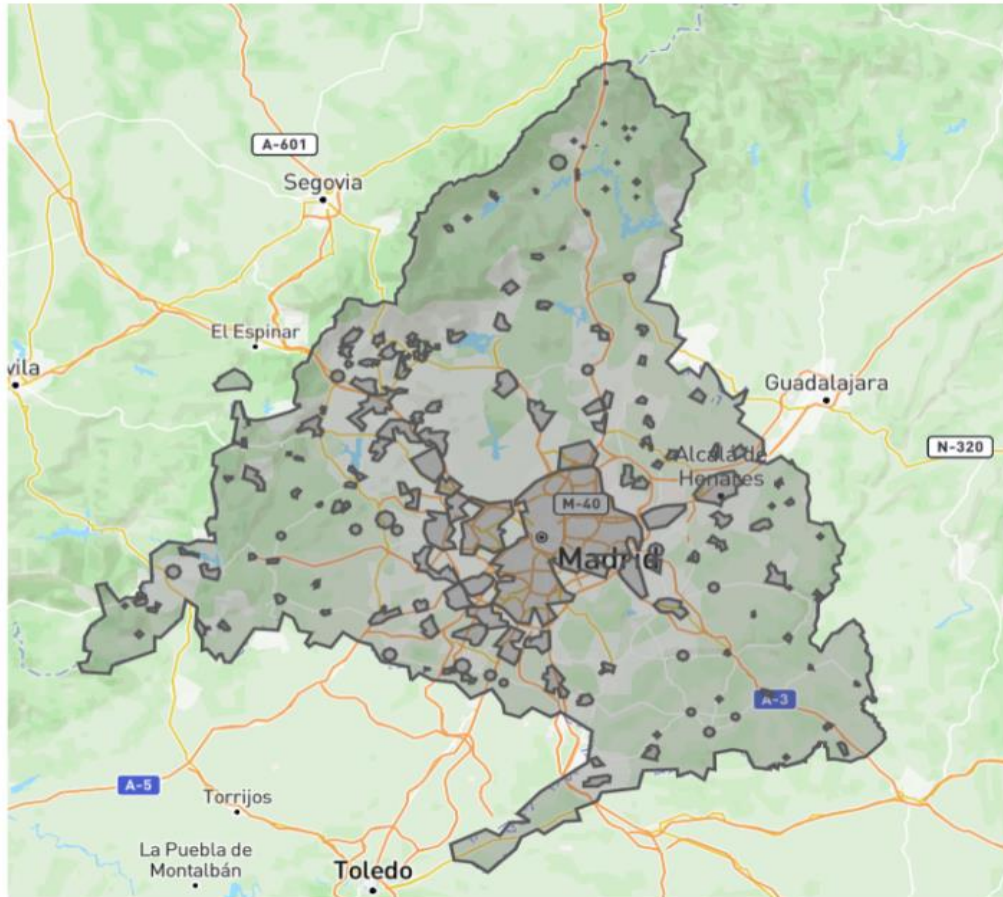
Captura de los costes calculados:

$$Coste = \left(94,64 + \frac{1233,33 + 900 + 965 + 25,02 + 1200}{82,9} \right) \cdot x \cdot 1,05$$

$$Coste = \left(94,64 + 165 + 257,67 + \frac{1233,33 + 900 + 965 + 25,02 * 3}{263700} \right) x \cdot 1,05$$

- Cálculo de superficies y división en zonas agrarias y residenciales: El equipo ha elaborado otra memoria donde explica la obtención de datos sobre las zonas de la comunidad de Madrid utilizando geojson.io. A partir de esos datos consiguió llevar a cabo el cálculo de superficies aproximado pedido.

Captura del mapa.



- Presupuesto final: A partir de la métrica de costes y de la división del mapa de la comunidad de Madrid el equipo explica en la memoria el presupuesto del que se parte para iniciar el proyecto.

Captura presupuesto final:

5. Resultados.

Tras los estudios realizados sobre la Comunidad de Madrid, se concluyen los siguientes datos:

- Superficie de la Comunidad de Madrid: 8.028 km²
- Superficie residencial de la Comunidad de Madrid: 1.036,85 km²
- Superficie agraria de la Comunidad de Madrid: 6.991,15 km²

Tras la división de la comunidad entre las zonas residencial y agraria podemos calcular cual es el área disponible para la instalación de las placas fotovoltaicas en cada zona.

- Superficie residencial disponible: 435,79 km²
- Superficie agraria disponible: 5.688 km²

Por último, aplicando la función de coste obtenida anteriormente se puede calcular el coste total de la operación.

- Presupuesto área residencial: **6,7500 x10¹⁰ €**
- Presupuesto área agraria: **3,1070 x10¹² €**
- Presupuesto total: **3,1745 x10¹² €**

En conclusión, el equipo de calculadora ha conseguido cumplir todos los requisitos propuestos para el primer sprint.

TRAMITADOR: Los requisitos del equipo son los siguientes:

- Acordar con los demás equipos el diseño de la página: El diseño de la página es algo distinto al diseño de resto de páginas por lo que se ha violado este acuerdo por parte del equipo. No se sigue el mismo diseño en las tablas, botones, desplegables y colores, es importante realizar el cambio. El logo de la página ahora mismo no es visible.

Captura de inicio de la página:

The screenshot shows the SKRM application interface. At the top is a blue header with the SKRM logo and navigation links: Inicio, Acerca de, Contacto, and Blog. Below the header, the main content area is titled 'Solicitudes'. It contains a table with 10 rows of request data. Each row has columns for Id Solicitud, Id Ayuda, Nombre, Apellido, Estado, and Más información. The 'Más información' column contains a button labeled 'Pulse para obtener más informacion'. At the bottom of the page, there is a footer with the text '© 2023 SKRM - Todos los derechos reservados'.

Id Solicitud	Id Ayuda	Nombre	Apellido	Estado	Más información
765	23	Nombre1	Apellido1	1	Pulse para obtener más informacion
865	56	Nombre2	Apellido2	1	Pulse para obtener más informacion
768	23	Nombre3	Apellido3	2	Pulse para obtener más informacion
790	23	Nombre4	Apellido4	2	Pulse para obtener más informacion
785	23	Nombre5	Apellido5	4	Pulse para obtener más informacion
907	20	Nombre6	Apellido6	3	Pulse para obtener más informacion
668	25	Nombre7	Apellido7	2	Pulse para obtener más informacion
293	17	Nombre8	Apellido8	1	Pulse para obtener más informacion
485	3	Nombre9	Apellido9	2	Pulse para obtener más informacion
947	12	Nombre10	Apellido10	4	Pulse para obtener más informacion

- Mostrar las peticiones de los ciudadanos: Tanto back como front han logrado cumplir este requisito con una buena conexión. La página a la que lleva para obtener información de la ayuda y poder aceptar y rechazar tampoco sigue lo estipulado en los contratos de front. No se puede realizar la búsqueda por estado de la petición. El código cumple con las buenas prácticas, pero se echan en falta comentarios.

Captura página información:

The screenshot shows the SKRM application interface for the 'AYUDA' section. It features a blue header with the SKRM logo and navigation links: Inicio, Acerca de, Contacto, and Blog. The main content area is titled 'AYUDA' and contains a form for requesting help. The form is divided into sections: 'DATOS CIUDADANO' (Citizen Data) with fields for Nombre and Apellidos; 'PETICIÓN' (Request) with fields for Identificador and Estado; and 'AYUDA' (Help) with fields for Identificador, Nombre, and Zona. Below the form, there is a section titled 'INFORMACIÓN GENERAL' (General Information) with a paragraph of text and a button labeled 'Ingresar su comentario'. At the bottom of the page, there is a footer with the text '© 2023 SKRM - Todos los derechos reservados'.

Identificador	Nombre	Zona
56	ayuda paneles solares para urbanizaciones	Pozuelo de Alarcón

Captura del swagger con las ayudas mock:

Code

Details

200

Response body

```
{
  "idAyuda": 56,
  "nombre": "ayuda paneles solares para urbanizaciones",
  "descripcion": "La ayuda paneles solares para urbanizaciones está disponible para las urbanizaciones...",
  "zona": "Pozuelo de Alarcon"
}
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Fri, 24 Feb 2023 20:38:42 GMT
keep-alive: timeout=60
transfer-encoding: chunked
vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
```

Captura swagger con peticiones mock:

Server response

Code

Details

200

Response body

```
[
  {
    "idSolicitud": 765,
    "idAyuda": 23,
    "nombre": "Nombre1",
    "apellido": "Apellido1",
    "estado": 1
  },
  {
    "idSolicitud": 865,
    "idAyuda": 56,
    "nombre": "Nombre2",
    "apellido": "Apellido2",
    "estado": 1
  },
  {
    "idSolicitud": 768,
    "idAyuda": 23,
    "nombre": "Nombre3",
    "apellido": "Apellido3",
    "estado": 2
  },
  {
    "idSolicitud": 790,
    "idAyuda": 23,
    "nombre": "Nombre4",
  }
]
```

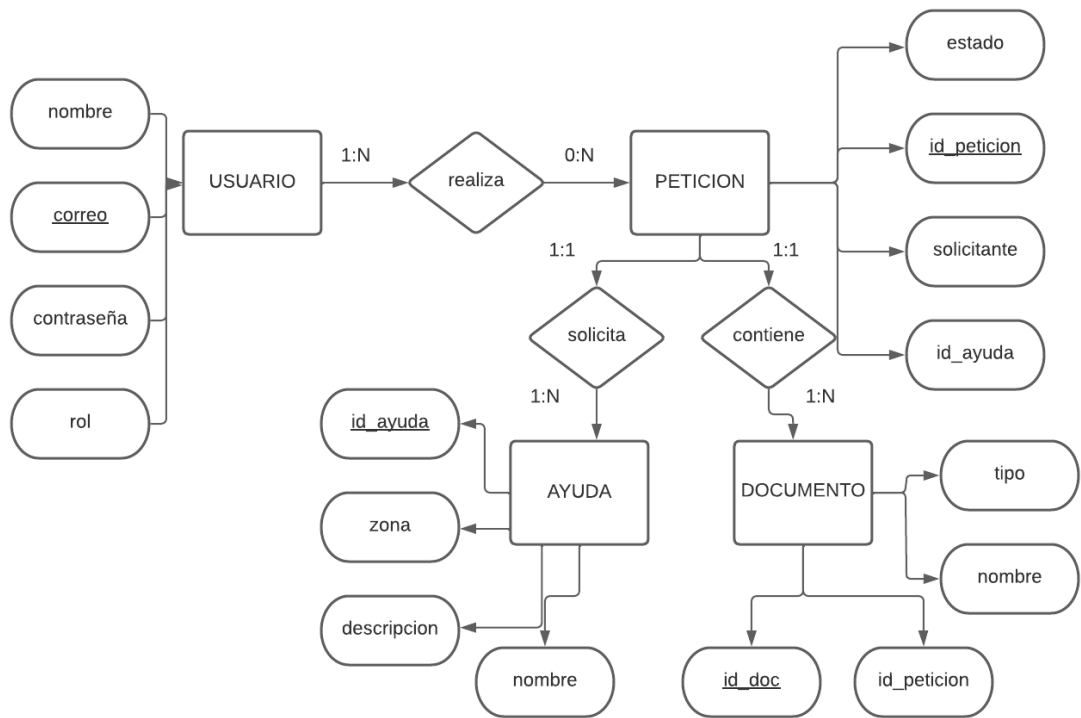
Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Fri, 24 Feb 2023 20:39:55 GMT
keep-alive: timeout=60
transfer-encoding: chunked
vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
```

BASE DE DATOS: Los requisitos del equipo para el primer sprint fueron los siguientes:

- Creación de un esquema entidad relación: El equipo ha creado el esquema adjuntado a continuación.



- Crear las tablas necesarias para implementar el esquema anterior: El equipo ha creado de manera satisfactoria las tablas de los 4 objetos que aparecen en el esquema.

main

document-management / src / main / java / skrm / persistence

History

Find file

Web IDE

↓

Clone

/ model / entities / +

Name	Last commit	Last update
..		
Ayuda.java	crear controller de peticion y de la clase peticionDatos	13 hours ago
Documento.java	Clases entidades usuario y ayuda	5 days ago
Peticion.java	cambios peticion, nombre columnas	13 hours ago
Usuario.java	Implementacion de Usuario	13 hours ago

Más concretamente se adjunta el código de la creación de la tabla ayuda. Como se puede ver, el código está comentado adecuadamente facilitando la comprensión del mismo.


```

33
34 //Constructor vacio
35 public Ayuda () {
36 }
37
38 /* Constructor de Ayuda
39 * Params:
40 * String nombre ->identificador
41 * String descripcion -> informacion de la ayuda
42 * String zona-> zona se aplica la ayuda
43 * Long presupuesto-> cuantía dedicada a la ayuda
44 */
45 public Ayuda (long idAyuda, String nombre, String descripcion, String zona, long presupuesto) {
46     this.idAyuda= idAyuda;
47     this.descripcion = descripcion;
48     this.nombre = nombre;
49     this.zona= zona;
50     this.presupuesto= presupuesto;
51 }
52
53 public Ayuda(AyudaDatos datos){
54     this.descripcion = datos.getDescripcion();
55     this.nombre = datos.getNombre();
56     this.zona = datos.getZona();
57     this.presupuesto= datos.getPresupuesto();
58 }
59
60 public long getIdAyuda() {
61     return idAyuda;
62 }
63
64 public String getNombre() {
65     return nombre;
66 }
67
68 public String getDescripcion() {
69     return descripcion;
70 }
71
72 public String getZona() {
73     return zona;
74 }
75
76 public long getPresupuesto() {
77     return presupuesto;
78 }
79 }

```

- Crear métodos para que el controlador pueda obtener información de la base: Han creado varios métodos para crear u obtener las diferentes ayudas, usuarios o peticiones.

Ejemplo del método createAyuda:

POST /createAyuda createAyuda	
Parameters Cancel	
Name	Description
ayudaDatos * required <i>(body)</i>	ayudaDatos Example Value Model <div> <pre>{ "descripcion": "Ayuda1", "nombre": "Pablo", "presupuesto": 1000, "zona": "Madrid" }</pre> </div>

Ejemplo del método getAyuda:



GET /getAyuda getAyudas

Request URL

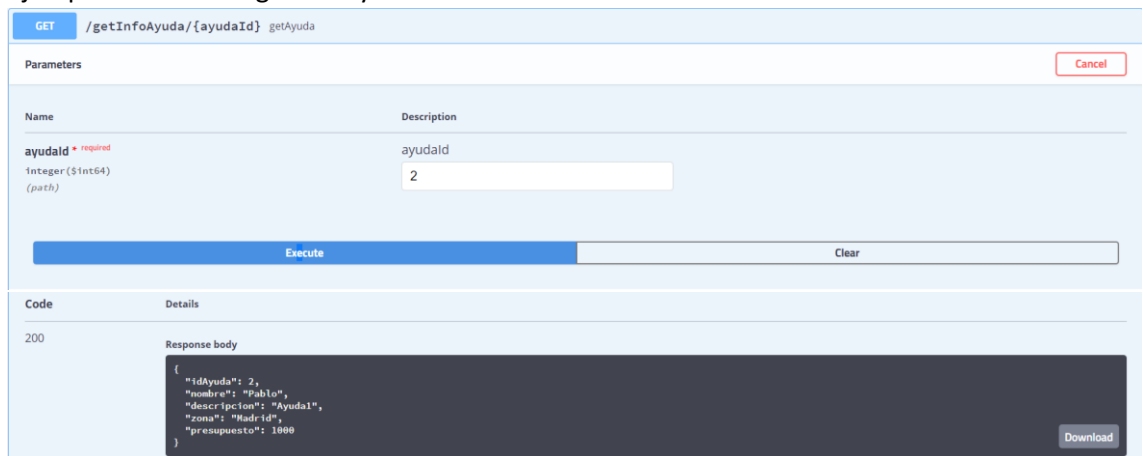
http://15ml.fi.upm.es:8088/getAyuda

Server response

Code	Details
200	<p>Response body</p> <pre>{ "idAyuda": 1, "nombre": "Segunda ayuda", "descripcion": "Esta es una descripción de la segunda ayuda", "zona": "Esta es la zona de la segunda ayuda", "presupuesto": 0 }, { "idAyuda": 2, "nombre": "Pablo", "descripcion": "Ayuda1", "zona": "Madr-id", "presupuesto": 1000 }</pre>

Download

Ejemplo del método getInfoAyuda:



GET /getInfoAyuda/{ayudaId} getInfoAyuda

Parameters

Cancel

Name	Description
ayudaId * required integer (\$int64) (path)	ayudaId 2

Execute Clear

Code	Details
200	<p>Response body</p> <pre>{ "idAyuda": 2, "nombre": "Pablo", "descripcion": "Ayuda1", "zona": "Madr-id", "presupuesto": 1000 }</pre>

Download

En resumen, la base de datos ha conseguido cumplir de manera satisfactoria todos los requisitos pedidos para este primer sprint.

CIUDADANO BACK: Los requisitos del equipo para el primer sprint fueron los siguientes:

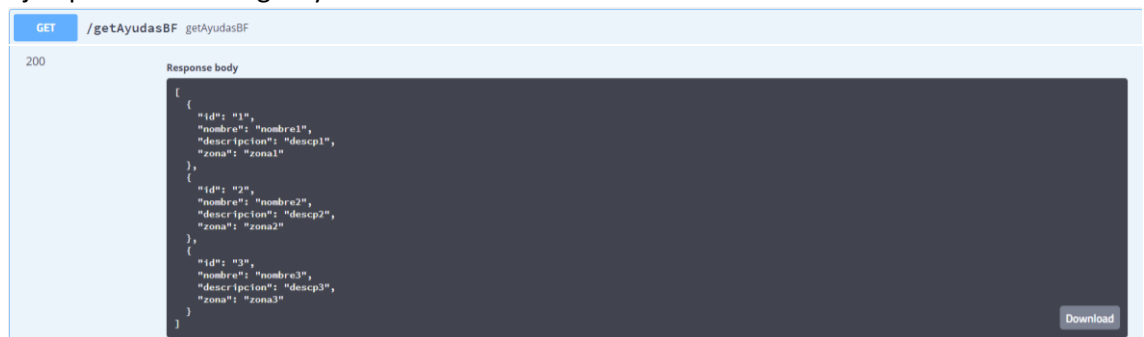
- Recibir las ayudas disponibles de la base de datos: Aunque durante el desarrollo del sprint ha habido problemas en la conexión de los diferentes equipos, han creado de manera local algunos casos de prueba para poder probar los códigos. Por ende, aunque no hayan conseguido conectarse con el controlador, sí que han creado correctamente los códigos pedidos. Solo una parte del código presenta comentarios.

```

19 @Service
20 public class APICaller {
21
22     @Autowired
23     private OkHttpClient client = new OkHttpClient() ;
24
25     public ciudadano getUserInfo(String email) throws IOException {
26         //Defino el url necesario para poder hacer la petición a controlador que me devuelva la informacion del usuario que tiene el email corres
27         String url = HttpUrl.parse("http://ismi.fi.upm.es:8086/getUserInfo").newBuilder().addQueryParameter("email",email).build().toString();
28         Request request = new Request.Builder().url(url).build();
29
30         Response response = client.newCall(request).execute();
31
32         //Si no se obtiene una respuesta valida
33         if (!response.isSuccessful())
34             System.out.println("ERROR, respuesta no valida");
35
36         //else -> tengo un json con la respuesta obtenida
37         JSONObject json = new JSONObject(response.body());
38         //Llamada a filterUserInfo
39         ciudadano ciu = filterUserInfo(json);
40
41         response.close();
42         return ciu;
43     }
44     public ArrayList<Ayuda> getAyudas() throws IOException {
45         Request request = new Request.Builder().url("http://ismi.fi.upm.es:8086/getAyudas").build();
46         Response response = client.newCall(request).execute();
47
48         if (!response.isSuccessful())
49             System.out.println("ERROR, respuesta no valida");
50
51         ArrayList<Ayuda> res = filterAyudas(response.body().toString());
52         response.close();
53
54         return res;
55     }
56 }

```

Ejemplo del método getAyudasBF:



GET /getAyudasBF getAyudasBF

200

Response body

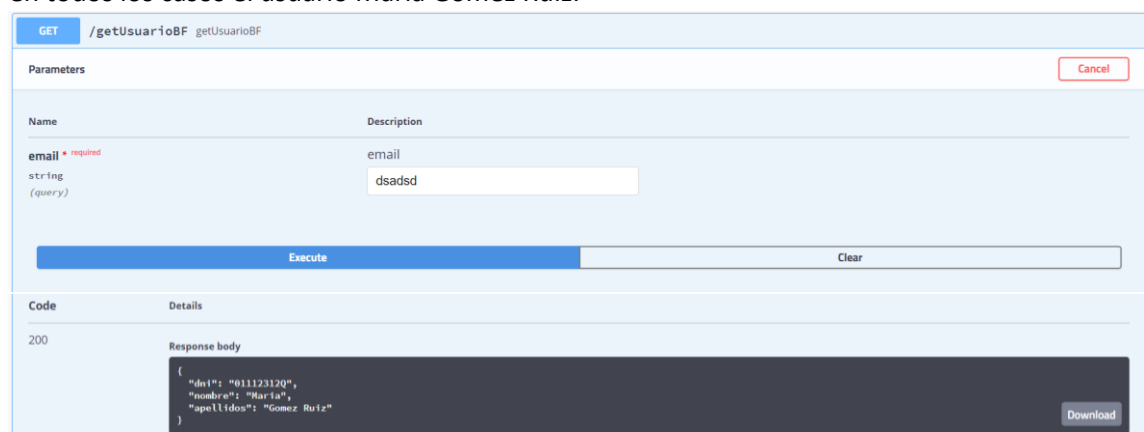
```

{
  "id": "1",
  "nombre": "nombre1",
  "descripcion": "descp1",
  "zona": "zona1"
},
{
  "id": "2",
  "nombre": "nombre2",
  "descripcion": "descp2",
  "zona": "zona2"
},
{
  "id": "3",
  "nombre": "nombre3",
  "descripcion": "descp3",
  "zona": "zona3"
}

```

Download

Ejemplo del método getUserInfoBF el cual, debido a problemas de conexión, devuelve en todos los casos el usuario María Gómez Ruiz:



GET /getUserInfoBF getUserInfoBF

Parameters

Cancel

Name	Description
email * required string (query)	email dsadsd

Execute Clear

Code Details

200

Response body

```

{
  "dni": "01112312Q",
  "nombre": "María",
  "apellidos": "Gomez Ruiz"
}

```

Download

- Establecer conexión con el front: Ambos equipos han creado varios contratos con la información que uno tiene que enviar al otro. Los métodos mostrados anteriormente han sido creados a partir de dichos contratos para el uso del front pero, por el momento, el equipo de Front no parece estar utilizándolos en su código.

- (Opcional) Filtrar la información recibida: Han creado un método que convierte el json devuelto por el controlador en un objeto del tipo “Ciudadano” para facilitar el uso de la información obtenida.

```
public ciudadano filterUserInfo(JSONObject json){
    Iterator<String> names = json.keys();
    ciudadano res = new ciudadano();
    res.setApellidos(json.getString(names.next()));
    res.setNombre(json.getString(names.next()));
    res.setDni(json.getString(names.next()));
    return res;
}
```

Por tanto, el equipo ciudadano Back ha conseguido cumplir de manera satisfactoria todos los requisitos obligatorios, e incluso los opcionales, pedidos para este primer sprint.

CIUDADANO FRONT: Los requisitos del equipo para el primer sprint fueron los siguientes:

- Firmar un contrato para llegar a un estilo común: Dicho contrato ha sido creado.

FRONT (pulmón, corazón, ojo y estómago)

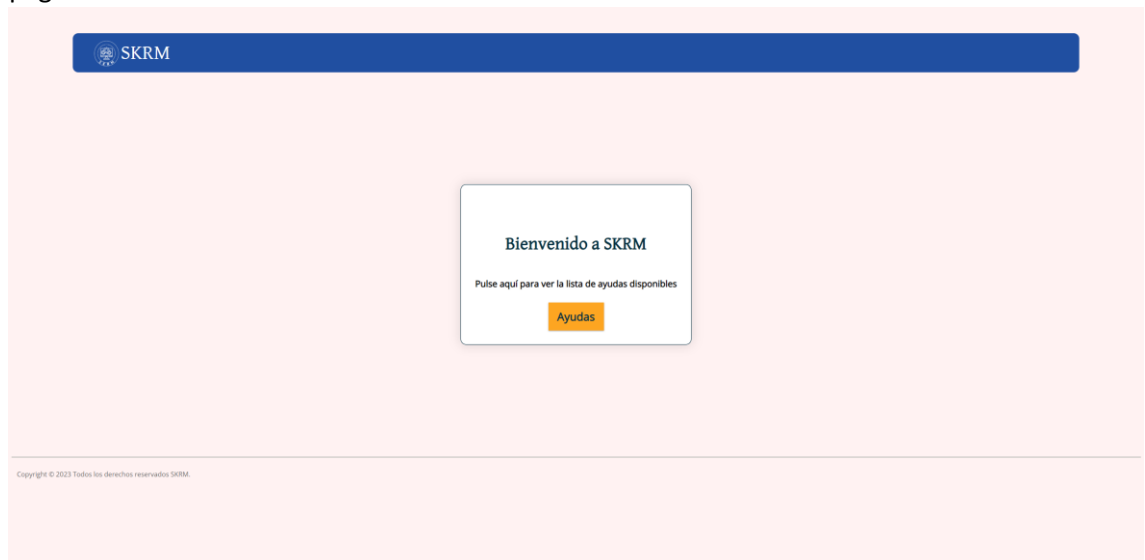
Se hará el front con CSS puro.

Se ha acordado:

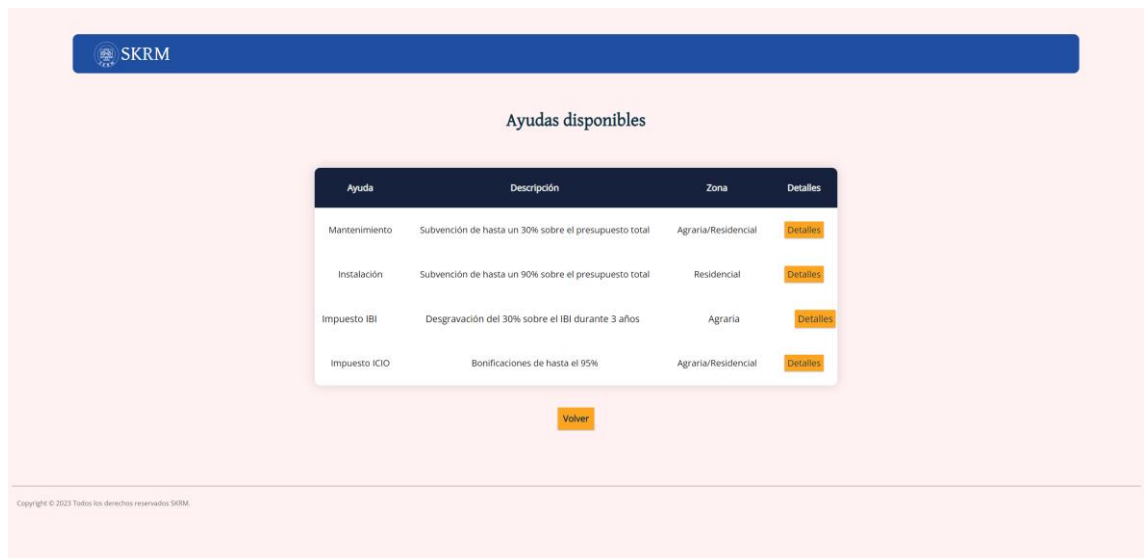
- logo arriba a la izquierda
- color azul (todos el mismo) que representa a la empresa (204Fa1)
- cajas bordeadas
- botones rectangulares
- fuentes gentium para títulos y subtítulos (fuente más pequeña)
- texto en open sans
- lista centrada en la página más o menos como una tabla en la que cada fila sea una solicitud con un botón a la derecha que se pueda pulsar para abrir para ver los datos en más detalle
- la barra de arriba con el logo ya de color azul
- la página principal irá en color azul agua muy claro
- las cajas de solicitudes irán en blanco normal
- el menú en forma de desplegable en forma de hamburguesa
- las cajas de aceptar o rechazar en el amarillo propuesto (puede ir una en claro y otro en oscuro)
- botón del presupuesto con un color diferente que pegue
- tenemos un footer común que incluya el nombre de la compañía, el año y el copyright (todos los derechos reservados).

- Creación de la página principal en la que ver las ayudas y documentos a subir en ellas: Han creado la página principal y han seguido el formato estipulado en el contrato mencionado anteriormente. Ahora bien no han creado la funcionalidad para que el usuario pueda ver más información sobre la ayuda y los documentos que este tiene que aportar. El código presenta comentarios para su mejor comprensión y la página es intuitiva de usar, cumpliendo las reglas de usabilidad propuestas. A continuación se adjuntan imágenes de la página.

Página principal: cabe mencionar que el footer no está alineado con el final de la página web.



Página secundaria: Una vez pulsado el botón “ayudas” que aparece en la página anterior se redirige al usuario a esta página. Recalcar de nuevo que los botones “Detalles” no muestran nada.



- Conectar el back con el front: Aunque han creado contratos y el equipo de back haya creado los métodos pedidos (también hay que decir que los ha creado a falta de muy poco tiempo para acabar el sprint) no los han utilizado en su código.

En definitiva, el equipo ciudadano Front no ha conseguido cumplir de manera satisfactoria algunos de los requisitos pedidos este primer sprint. Cabe mencionar que, aunque no hayan cumplido todos los requisitos, han conseguido crear una página funcional que seguía el estilo propuesto.

SUPERVISOR: Los requisitos del equipo para el primer sprint fueron los siguientes:

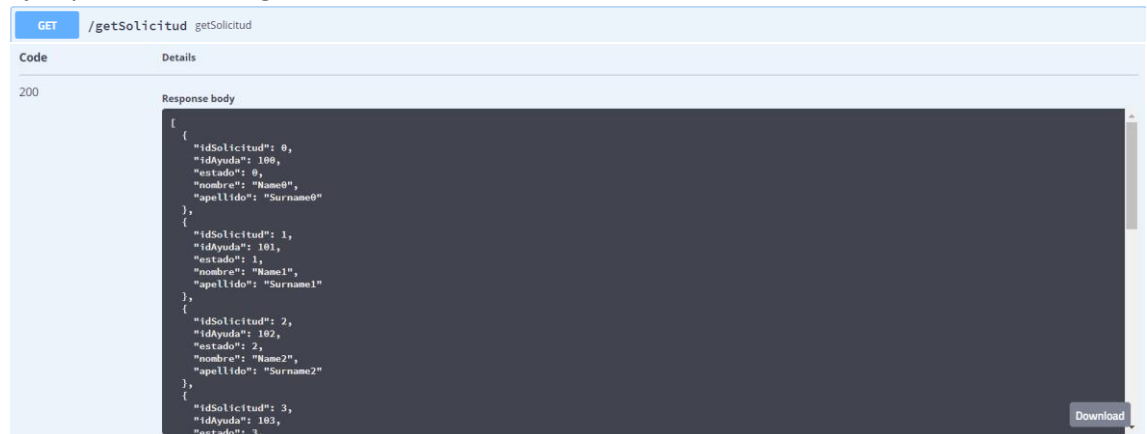
Back-End:

- Conectar el front con el back: Han creado tres métodos relacionados con el funcionamiento de su front. Ahora bien, debido a problemas en la conexión entre los equipos las pruebas de estos métodos se harán en base de códigos de prueba suministrados por los propios programadores. Todos los métodos mencionados anteriormente parecen funcionar de manera correcta con los casos de prueba suministrados por los programadores. No todo el código está correctamente comentado.

```
@GetMapping("/ayudasFront")
public ResponseEntity<List<Ayuda>> ayudas(@RequestParam(name="idAyuda" ,required = false, defaultValue="-1") int idAyuda) throws IOException {
    //List<Ayuda> serviceResponse = apiCaller.ayudaAPI(idAyuda);
    List<Ayuda> serviceResponse=new ArrayList<>();
    if(idAyuda != -1){ // Si no nos dan una id de ayuda, como las apis de los controladores siguen sin ser disponibles generamos una nueva
        serviceResponse.add(new Ayuda(idAyuda, "ayuda"+idAyuda, "descripcion de la ayuda " + idAyuda, "Madrid zona " + idAyuda));
        return new ResponseEntity<>(serviceResponse, HttpStatus.OK);
    }
    serviceResponse.add(new Ayuda(1, "ayuda1","descripcion1","Madrid"));
    serviceResponse.add(new Ayuda(2, "ayuda2","descripcion2","Badajoz"));
    serviceResponse.add(new Ayuda(3, "ayuda3","descripcion3","Toledo"));
    return new ResponseEntity<List<Ayuda>>(serviceResponse,HttpStatus.OK);
}

@GetMapping("/presupuestoFront")
public ResponseEntity<Integer> presupuesto() throws NumberFormatException, IOException {
    //int serviceResponse =apiCaller.presAPI();
    int serviceResponse=5000;
    return new ResponseEntity<Integer>(serviceResponse,HttpStatus.OK);
}
```

Ejemplo del método getSolicitud:



GET /getSolicitud getSolicitud

Code Details

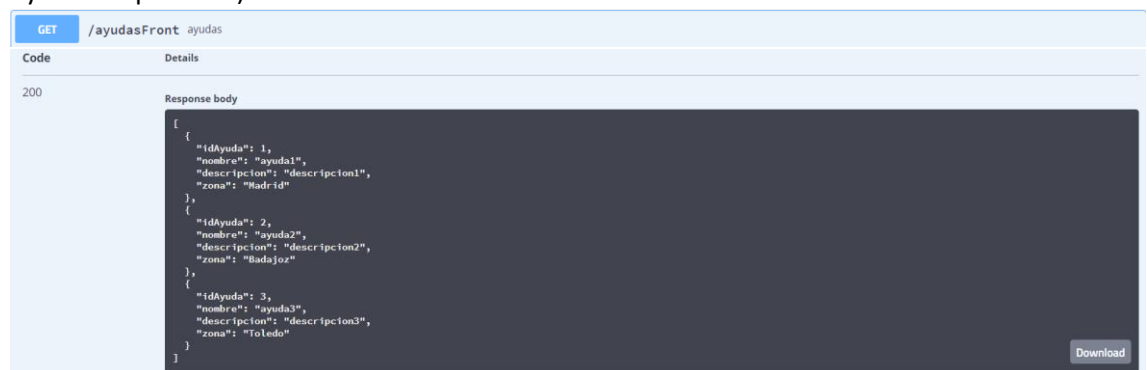
200

Response body

```
[
  {
    "idSolicitud": 0,
    "idAyuda": 100,
    "estado": 0,
    "nombre": "Name0",
    "apellido": "Surname0"
  },
  {
    "idSolicitud": 1,
    "idAyuda": 101,
    "estado": 1,
    "nombre": "Name1",
    "apellido": "Surname1"
  },
  {
    "idSolicitud": 2,
    "idAyuda": 102,
    "estado": 2,
    "nombre": "Name2",
    "apellido": "Surname2"
  },
  {
    "idSolicitud": 3,
    "idAyuda": 103,
    "estado": 3
  }
]
```

Download

Ejemplo del método ayudasFront con el parámetro -1 (el cual devuelve todas las ayudas disponibles):



GET /ayudasFront ayudas

Code Details

200

Response body

```
[
  {
    "idAyuda": 1,
    "nombre": "ayuda1",
    "descripcion": "descripcion1",
    "zona": "Madrid"
  },
  {
    "idAyuda": 2,
    "nombre": "ayuda2",
    "descripcion": "descripcion2",
    "zona": "Badajoz"
  },
  {
    "idAyuda": 3,
    "nombre": "ayuda3",
    "descripcion": "descripcion3",
    "zona": "Toledo"
  }
]
```

Download

Ejemplo del método presupuestoFront:

GET /presupuestoFront presupuesto	
Code	Details
200	<div>Response body</div> <div>5000</div> <div>Download</div>

- Crear contratos con el equipo controlador: El equipo de supervisor ha firmado varios contratos con el equipo controlador referentes a los formatos de la información recibida.
























Front-End:

- Seguir un diseño pactado con los demás grupos: Han firmado un contrato con el resto de grupos referente al estilo a seguir, pero no han utilizado el footer común en el resto de páginas si no uno propio.


2023 - Skrm





















- Mostrar peticiones en fase final: Han creado una página web funcional y que cumple los requisitos pedidos. Además, han creado una barra para la búsqueda de las peticiones, aunque esta no es muy intuitiva debido que no indica en función de que parámetro filtra las peticiones mostradas y tiene una fuente demasiado pequeña para poder leer bien el texto que aparece en ella. El diseño de la página es fácil de entender, pero en la página “más info” la fuente utilizada para la información de las ayudas es un poco pequeña y el método para volver a la página anterior es poco intuitivo para la mayoría de usuarios (podrían poner un botón más claro o que, al menos, se pueda volver atrás pulsando el nombre de la empresa y no solo el logo).

Página principal del front:


 SKRM 			
Presupuesto Restante: 5000			
Buscar petición 			
Petición	Nombre	Más Info	Estado
0	Name0 Surname0		
1	Name1 Surname1		
2	Name2 Surname2		
3	Name3 Surname3		
4	Name4 Surname4		
5	Name5 Surname5		
6	Name6 Surname6		
7	Name7 Surname7		
8	Name8 Surname8		
9	Name9 Surname9		


Ejemplo del uso del buscador de peticiones:

5 

Petición	Nombre	Más info	Estado
0	Name0 Surname0		
1	Name1 Surname1		
2	Name2 Surname2		
3	Name3 Surname3		
4	Name4 Surname4		
5	Name5 Surname5		
6	Name6 Surname6		
7	Name7 Surname7		
8	Name8 Surname8		
9	Name9 Surname9		

Página secundaria a la cual te redirige la web cuando pulsas el botón azul de la columna “más info”:

 SKRM


Supervisor

INFORMACIÓN

Nombre

Alfonso

Apellidos

Indares Irujo

PETICIÓN

Id

1

Zona

Madrid

CAUSA

descripcion

Aceptar

Denegar

Proponer

2023 - SKRM

all rights reserved

En conclusión, el equipo supervisor ha conseguido cumplir de manera satisfactoria los requisitos pedidos para este primer sprint.

OCR: los requisitos pedidos para el primer sprint son los siguientes:

- Establecer una conexión con el back de CIUDADANO para el intercambio de un pdf o formulario: Se ha creado una función denominada “upload_pdf()” la cual se encarga de recibir un documento de tipo pdf e imprimirlo por pantalla dando el mensaje ‘Archivo recibido’ si la acción se ha realizado de forma correcta o el mensaje ‘Archivo no recibido’ si ha ocurrido un error en el proceso.

La siguiente captura muestra dicha función, cabe recalcar la ausencia de comentarios descriptivos que permitan una mejor lectura del código:

```
def upload_pdf():  
    file = request.files.get('file')  
    if file:  
        doc = fitz.open(stream=file.read(), filetype="pdf")  
        text = pdf2text(doc)  
        print(text)  
        return 'Archivo recibido'  
    else:  
        return 'Archivo no recibido'
```

- Convertir el pdf recibido a tipo txt: Para este caso se nos presenta una función llamada “pdf2text(doc)” la cual recibe un documento como argumento y se encarga de procesarlo al formato txt. Sin embargo, el uso de la función “get_text()” no está definida y no permite su comprobación correcta. Además, no se realiza ningún tipo de control de seguridad para verificar si el doc recibido es de tipo pdf o es de otro tipo. En la siguiente captura se muestra la función e igual que antes carece de comentarios descriptivos del código:

```
def pdf2text(doc):  
    text = ""  
    for page in doc:  
        text += page.get_text()  
    return text
```

Esta función en particular ha recibido algunos cambios ya que en previas versiones sí realizaba la tarea designada, como se muestra en la siguiente captura con el uso de un pdf llamado “testing_group_v2.pdf” como ejemplo:

```
from pdfminer.high_level import extract_text  
  
# -*- coding: utf-8 -*-  
  
#Función que mete en un string el texto de un pdf  
def pdf2text(pdf):  
    text=extract_text(pdf)  
    return text
```

```
pdf2text("testing_group_v2.pdf")
```

' Dado que esto es algo que no se ha visto antes, debéis ver las siguientes líneas como una\n\nexplicación y no como una guía estricta. Sois libres de proponer cambios o propuestas que mejoren\n\nesta idea.\n\n2/12/2022\n\nVisión general equipo de testing\n\nLas labores del equipo son dos:\n\n1. Testear la aplicación y certificar su calidad\n\n2. Generar datos\n\nLa parte de generación de datos es secundaria y debemos verla como un tipo de subcontrata con otros\n\nequipos de la empresa en base a lo que vayan necesitando. Veamos entonces una introducción a los dos\n\npuntos desde los que abordaremos el testing.\n\nTests end-to-end (e2e) basados en los requisitos del cliente\n\nEstos tests los llevaremos a cabo interactuando con las interfaces gráficas que verán todos los usuarios de la\n\naplicación y verificando, en un primer momento de forma manual, el correcto funcionamiento de todo el\n\nsistema acorde a los requisitos de Andrés. La definición de "correcto funcionamiento" es tarea nuestra y la\n\nllevaremos a cabo desmenuzando los requisitos en pequeñas partes para las cuales pensaremos tests que\n\ncomprueben los diversos resultados posibles por orden de prioridad.\n\nAuditoría interna de metodología testing\n\nPara esta parte, comprobaremos que los procesos internos de testing de cada grupo se están llevando a cabo\n\nde forma correcta y qué, por tanto, sirven como evidencia del buen funcionamiento del sistema. De nuevo,\n\nntenemos que definir qué es bueno y qué no. Para esto, dividiremos el problema en pequeñas partes\n\nnevaluables. Con los datos que recolectemos sobre los equipos elaboraremos un report que presentaremos a\n\nntoda la empresa al final de cada sprint para poder corregir fallos. Este documento se lo presentaremos\n\ntambién al cliente en las demos.\n\nTest e2e\n\nEstos tests serán manuales en un primer momento. Por tanto, pondremos el foco en hacer tests de calidad y\n\ndetallados.\n\nMétodo SMART\n\nAntes de contaros mi propuesta sobre como llevar estos tests a cabo, veamos el concepto SMART (Specific\n\nMeasurable Achievable Relevant Timely). Esta es una regla nemotécnica usada a la hora de crear tareas,\n\nobjetivos o hábitos. Y la implementación a nuestra manera de testear es la siguiente:\n\nTestearemos casos específicos. Tantos como sean necesarios pero de forma concreta y precisa\n\nDefiniremos una métrica de éxito\n\nTestearemos objetivos razonables y dentro del scope del proyecto\n\nTendremos en cuenta las fechas de entrega y deadlines varias\n\n1 / 2\n\n2/12/2022\n\nMétodo de trabajo\n\nTraduciremos los requisitos del cliente en pequeñas frases concisas de tal forma que podamos decir si se\n\ncumplen o no con un simple sí/no.\n\nTendremos un documento con la lista de todos estos requisitos "simplificados" y los iremos expandiendo o\n\nmodificando en función de las opiniones del cliente (hasta cierto punto). Estos tests los tendremos diseñados\n\nnantes de que se desarrollen y una vez estén en funcionamiento los probaremos y documentaremos el\n\nproceso. Dentro de este diseño, pondremos ejemplos concretos de cosas a probar.\n\nAuditoría de metodología\n\nNuestro objetivo como testers es dar a Andrés la confianza necesaria en nuestro proyecto. Para ello,\n\nndebemos definir de la forma más objetiva posible qué es "buen funcionamiento". Para esto tenemos\n\nmúltiples enfoques, yo os propongo el siguiente.\n\nTendremos un sistema de puntuación de tal forma que daremos una nota por cada categoría y\n\npromediaremos la nota del equipo en su conjunto. Las categorías a vigilar son las siguientes:\n\n1. Seguimiento de la metodología\n\n2. Evidencias por parte de los desarrolladores\n\n3. Calidad y profundidad del testing realizado a nivel de ticket\n\n4. Valoración de los tests del líder\n\n0s propongo también elegir un equipo aleatorio cada sprint y dedicarle una atención especial probándolo en\n\ndetalle. Esto es una idea adaptada de un tipo de testing que hacen compañías como Netflix con Chaos\n\nMonkey, donde un bot va tirando servicios de la aplicación aleatoriamente para testear la resiliencia de sus\n\nsistemas.\n\nReporting\n\nCon todos estos datos recopilados por los miembros del equipo, nuestro líder elaborará un report donde se\n\nndarán datos globales sobre la calidad del testeo interno que se está realizando en la aplicación y sobre\n\nnuestros propios tests e2e y cuantos de ellos pasan en el momento de elaborar el report. Estos reports\n\npodrían estar ubicados en el repositorio del equipo para poder exponerlos a toda la empresa si hiciese falta\n\nen el futuro.\n\n2 / 2\n\nx0c'

- Conexión con el Controlador de peticiones: para este requisito se ha hecho uso de una serie de mandatos que permiten establecer la conexión pedida. Por tanto, se concluye que se cumple con lo establecido de forma correcta.
- Se adjunta capturas que muestran la conexión:

```
app = Flask(__name__)
app.config['DEBUG'] = True
```

```
@app.route('/enviar-documento', methods=['POST'])
```





```
app.run(host='0.0.0.0')
```

```
* Serving Flask app '__main__'
* Debug mode: on

WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://138.100.242.97:5000
Press CTRL+C to quit
* Restarting with stat
```



CONTROLADOR: los requisitos planteados para este primer sprint son los siguientes:

- Familiarización con las tecnologías del proyecto: para este requisito se ha comprobado que los programadores/integrantes encargados de esta parte han reflejado sus avances en el desarrollo del código para que realice correctamente las funciones pedidas. Por ello, se puede observar que, efectivamente, ha habido un trabajo detrás el cual se verá en las siguientes capturas que dan prueba de ello:

Name	Last commit	Last update
..		
 ExampleController....	He corregido los fallos de la marge	14 hours ago
 HealthController.java	Basic back document-controller	2 weeks ago
 SolicitudController....	Gestión de errores de peticiones http	3 days ago
 userController.java	Método acceso	14 hours ago

Aquí se observa que realizaron modificaciones a su código con el objetivo de intentar subsanar los fallos cometidos anteriormente relacionados con la conexión entre las distintas partes.

Además, el conjunto de Merge Request (en algunos casos no se ha seguido la metodología pedida) demuestra su esfuerzo por familiarizarse con el trabajo, como se puede observar en la siguiente captura:

Clase userController con el método acceso !4 · created 14 hours ago by RICARDO ALBERTO FERREIRO DE AGUIAR	MERGED  0 updated 14 hours ago
Rf 4 !3 · created 15 hours ago by RICARDO ALBERTO FERREIRO DE AGUIAR	MERGED  0 updated 14 hours ago
Rf 4 !2 · created 1 day ago by RICARDO ALBERTO FERREIRO DE AGUIAR	 CLOSED  1 updated 1 day ago
Métodos getSolicitud y getUserInfo !1 · created 3 days ago by CARLOS RODRÍGUEZ LIMÓN	MERGED   8  Approved  0 updated 3 days ago

- Asegurar el cumplimiento de requisitos de los demás grupos: para este apartado se observa que no se ha acabado de cumplir del todo ya que en algunos casos no se han seguido las normas de estética de algunos front, mientras que otros grupos sí la han cumplido. Por otra parte, el cumplimiento general de requisitos se puede dar como satisfactorio debido a que gran parte de los mismo se han realizado de forma correcta.

Por último, se incluyen una serie de capturas de su swagger para mostrar sus avances a la hora de asegurar conexiones tanto con la Base de Datos como con el resto de componentes del proyecto:

POST

/send-db sendDb

Name	Description
datos <small>* required</small> (body)	datos <div> <div>Example Value</div> <div>Model</div> </div> <pre>{ "discord": "string", "email": "string", "matricula": "string", "nombre": "string", "password": "string" }</pre> <div> Parameter content type <div>application/json</div> </div>

Responses

Response content type

/

Code	Description
200	<div>OK</div> <div> <div>Example Value</div> <div>Model</div> </div> <pre>{ "cadena": "string" }</pre>

En los siguientes ejemplos no se puede verificar el funcionamiento correcto debido a la falta de indicaciones en el código para su correcto testeo y debido a la poca claridad en sus swagger para su comprobación:

GET

/acceso/{userId}/{password} acceso

Name	Description
password <small>* required</small> string (path)	password
userId <small>* required</small> string (path)	userId

Responses

Response content type

/

Code	Description
200	<div>OK</div> <div> <div>Example Value</div> <div>Model</div> </div> <pre>{ "first": 0, "second": "string" }</pre>

GET

/getUserInfo/{userId}getUserInfo

Name	Description
userId * required string (path)	userId

Responses

Response content type */*

Code	Description
200	<div>OK</div> <div><div>Example Value Model</div><div><pre>{ "apellido1": "string", "apellido2": "string", "dni": "string", "nombre": "string"}</pre></div></div>

En resumen, el equipo de Controlador ha cumplido parte de sus requisitos pedidos, pero debe seguir trabajando a la hora de asegurar las conexiones entre las distintas partes del proyecto ya que depende mucho de su correcto funcionamiento.