

# Firmadocs

Firmar y enviar documentos en papel es (o debería ser) algo del pasado: además de tener que usar papel e imprimir, hay que firmarlo a mano y enviarlo por correo postal. ¡No sólo es tedioso sino costoso y poco ecológico! Y ni hablar cuando el documento tiene que ser firmado por personas que están a grandes distancias. Por eso creamos Firmadocs, una plataforma en español de firma electrónica en la que cualquier persona puede subir sus documentos, completarlos y firmarlos colaborativamente. No importa si se trata de un contrato, una carta de recomendación, o una declaración jurada, ¡Firmadocs es para vos!



Así es como Dani, de Firmadocs, comenzó a presentar con mucha pasión su producto al equipo de la consultora 2Diseños, que se encargará de desarrollarlo. Aunque la sala no quedó tan emocionada, se apresuró a preguntar sobre los requerimientos.

## Contexto

### Documentos

En Firmadocs podés iniciar y participar en procesos de firma colaborativa. Para comenzar un proceso de firma colaborativa de un documento, vas a necesitar primero subir un PDF (el documento en cuestión) y luego compartirlo, agregando colaboradores al mismo.

Cuando agregás un colaborador<sup>1</sup>, tenés que especificar si se trata de un firmante o un lector. Los lectores **deberán** ingresar al documento para leerlo e indicar que lo hicieron, mientras que los firmantes **deberán** cargar sus datos y firma. Es importante que debe quedar registro de quienes completaron su acción correspondiente; si alguno de los colaboradores no lo hizo, el proceso de firma no puede ser finalizado.

*Configurar comportamiento*

Además, antes de comenzar a agregar colaboradores, deberás indicar si firmarán o leerán en orden indistinto u orden prefijado.

- Orden indistinto: cada colaborador recibe la notificación al mismo tiempo y puede realizar su acción a partir de ese momento;
- Orden prefijado: cada colaborador recibe la notificación en orden. Un colaborador es notificado y puede realizar su acción recién cuando el anterior ha concluido la suya.

Una vez que el proceso de firma haya sido configurado, con todos sus colaboradores asignados, podrás liberar el documento, notificando a los colaboradores que corresponda.

A medida que cada colaborador sea notificado, podrá ingresar al sistema, buscar el documento entre los que le fueron compartidos, y realizar su acción. No podrá realizar otra que no le corresponda, ni antes de que le sea habilitada. Tampoco podrá ver los documentos compartidos si aún no es su turno.

Cuando el último colaborador complete su acción, el proceso de firma será marcado automáticamente como finalizado.

<sup>1</sup> La persona que inicia el proceso de firma también puede asignarse a sí misma como colaboradora, típicamente para firmar el documento

## Notificaciones

Ramón, Agustín  
07 (nueve)

K31



Es importante que todas las personas sean notificadas a través de medios que elijan (correo electrónico, Whatsapp, notificación en aplicación, SMS, entre otros, configurables por usuario) sobre los distintos eventos que van ocurriendo, incluyendo:

- Cuando se te comparte un documento *owner!*
- Cuando un colaborador de un proceso de firma que iniciaste completa su acción
- Cuando un proceso de firma que iniciaste culminó
- **Diariamente**, cuando sos un colaborador y no completaste tu acción.

Cron?

## Firma electrónica

Cada vez que un firmante firma un documento, al mismo debe anexarse un código hexadecimal único que representa su firma electrónica y demuestra su validez. Este código debe ser generado usando un componente externo<sup>2</sup> en ese preciso momento, ni antes ni después.

 <b>GeneradorDeFirma</b> 
<code>generarFirma(nombre, apellido, email, telefono) : String</code>

Adaptación de integraciones

## Planes de facturación

Firmadocs permite a los usuarios contratar un plan acorde a sus necesidades: se cuenta con planes plata, bronce y oro, que permiten iniciar hasta 5, 20 y 100 procesos de firma mensuales. Una vez que se agotó el plan, **no se debe permitir generar nuevos procesos de firma más hasta el mes siguiente.**

## Requerimientos detallados

1. Como iniciador, deseo poder iniciar el *proceso de firma* de un documento, indicando el PDF.
2. Como iniciador, deseo poder especificar si el *proceso de firma* tendrá orden o será de orden indistinto.
3. Como iniciador, deseo poder asignarle colaboradores a un *proceso de firma*, especificando si serán lectores o firmantes.
4. Como iniciador, deseo poder liberar un *proceso de firma*, enviando notificaciones a cada uno de los colaboradores, en orden (o a todos a la vez, si es orden indistinto).
5. Como colaborador, deseo poder listar los *procesos de firma* que tengo compartidos.
6. Como iniciador o colaborador, debe poder listar las personas que leyeron el documento al momento actual del *proceso de firma*
7. Como iniciador o colaborador, debe poder listar las personas que firmaron el documento al momento actual del *proceso de firma*
8. Como firmante, deseo poder firmar un documento del que sea colaborador de su *proceso*.
9. Como lector, deseo poder marcar como leído un documento del que sea colaborador de su *proceso*
10. Como colaborador, deseo recibir recordatorios cuando no haya realizado mi acción.
11. Como ~~colaborador~~ **iniciador**, deseo poder anular un *proceso de firma* no liberado. A nivel facturación del plan no se contabilizará.
12. Como ~~colaborador~~ **iniciador**, deseo poder anular un *proceso de firma* ya liberado. A nivel facturación del plan sí se contabilizará.

<sup>2</sup> En la realidad esto es bastante más complejo e involucra cuestiones criptográficas y el ingreso de una imagen.

Ranieri, Agustín

Primeros notos  
01 (nueve)

K3A

1. Es un problema creacional, se resuelve instanciando un proceso de firma.
2. Como ocurre al principio, no se puede cambiar y tiene comportamiento distinto dependiendo de uno u otro, me da a entender que puedo usar herencia para resolverlo ~~o sea tiene un comportamiento genérico de agregar gestos de los pautados~~.  
Si es en orden, es probable que necesite que cada colaborador conozca al siguiente para notificarlo. (Posible Iterator)

3. El procedimiento en la asignación puede diferir si es en orden o no:
  - si es en orden, al colaborador anterior se le asigna un siguiente a notificar
  - si es sin orden, <sup>directamente le</sup> directamente lo agregamos a una lista.

4. Ambos tipos de procesos tendrán una forma distinta de ser liberados:
  - en el primer caso (en orden), se notifica al primero (y se le habilita)
  - en el segundo caso, se hace un foreach para notificar a todos (y habilitar)

Ambos tipos de colaboradores tendrán un usuario, que en el caso de la lectura se agregarán como leídos y en el caso de la firma usarán sus datos para generarla.

5. Es simplemente contar con un Repositorio de procesos de firma y poder filtrarlos según si son del usuario dueño.

7y 6. Los usuarios lectores se van agregando a medida que completan la acción, al igual que las firmas de los usuarios firmantes



8. y 9. Ambos me dan a entender la idea de cosificar comportamiento :  
crear una acción que puede ser ejecutada más adelante.  
(En el caso de la firma es el más claro de todos porque se genera en  
el momento sí o sí).

10. La colaboración puede estar en estado deshabilitada, pendiente o  
completada. ~~De esta forma~~ De esta forma, se podrían filtrar ~~los~~  
~~los~~ los que sean pendientes.

11. y 12. Es trivial, lo resuelvo con un booleano.



1. abstract class ProcesoDeFirma {  
 colaboraciones : Colaboracion = []  
 iniciador : Usuario

estaActivo : boolean = true

lectores : Usuario [] = []

firmas : Firma [] = []

documento : File

estaLiberado : boolean = false

fechaGeneracion = LocalDate.now()

}

2. new ProcesoConOrden (iniciador, documento);

new ProcesoSinOrden (iniciador, documento);

Arrancan en estado  
DESABILITADA

3. # ProcesoConOrden . agregar (colaboracion) {

this.colaboraciones.last().setSiguiente(colaboracion)

super.agregar(colaboracion)

}

podría aprovechar  
que es un List

# ProcesoDeFirma . agregar (colaboracion) {

this.colaboraciones.add(colaboracion)

}

4. # ProcesoConOrden . liberar () {

this.colaboradores.getFirst().habilitar()

this.liberado = true;

}

habilitarEn(this)

# ProcesoSinOrden . liberar () {

this.colaboradores.forEach({ it => it.habilitar() })

this.estaLiberado = true;

}

duplicado  
=> template

(sigue en sig. hoja)

```

4. # Colaboracion.habilitarEn(proceso) {
  (cont.)  ✓ this.usuario.notificarCompartido(proceso)
           this.estado = PENDIENTE;
           }

```

```

# Usuario.notificarCompartido(proceso) {
  ✓ this.notificadores.forEach({ it => it.notificarCompartido(proceso, this) })
}

```

```

5. # RepositorioProcesos.getAllByUsuario(usuario) {
  return this.procesos.filter({ it => it.tieneUsuario(usuario) })
}

```

```

# ProcesoDeFirma.tieneUsuario(usuario) {
  return this.colaboraciones.any({ it => it.tieneUsuario(usuario) })
}

```

Ok, aunque falta detalle  
porque depende del  
modo de colabora

```

6. # ProcesoDeFirma.getLectores()

```

```

7. # ProcesoDeFirma.getFirmantes() {
  return this.firmantes.map(Firmante::getUsuario);
}

```

```

8. # Firma.realizarEn(proceso) {
  super(proceso) (ver punto 9)

```

```

  ✓ proceso.agregarFirma(

```

```

    this.firmador.generar(usuario)
  )

```



```

# Firmador . generar (usuario) {
    return new Firmante (usuario,
        this.generator . generarFirma (
            usuario.nombre, usuario.apellido, usuario.email, usuario.telefono
        )
    );
}

```

```

# Proceso . agregarFirmante (firmante) {
    this.firmantes.add(firmante);
    this.iniciador . notificarFirma (firmante);
}

```

```

9. # Lectura . realizarEn (proceso) {
    super (proceso);
    proceso . agregarLector (this.usuario);
}

```

```

# Proceso . agregarLector (usuario) {
    this.lectores.add(usuario);
    this.iniciador . notificarLectura (usuario);
}

```

```

# Colaboracion . realizarEn (proceso) {
    this.estado = COMPLETADA;
    if (this.siguiente != NULL) {
        this.siguiente . habilitarEn (proceso);
    }
}

```

Es, hay dependencias implícitas entre el orden y esto

10. Poner un cron en el sistema operativo que cada cierto periodo ejecute desde ahí un ejecutable ya compilado, con el siguiente método main():

```

main() {
    RepoProcesos . getInstance().getAll().forEach({ it => it . notificarPendientes() })
}

```

↳ Filtro si ya estan en

```

# ProcesoDeFirma . notificarPendientes () {
    this.colaboraciones . forEach({ it => it . notificarPendiente (this) })
}

```

solo registros



10. ... (continuación, ver hoja 3/4)

4/4

```
# Colaboracion. notificarPendiente(proceso) {  
  if (this.estado == PENDIENTE) {  
    this.usuario.notificarPendiente(proceso)  
  }  
}
```

Y luego, el mismo procedimiento que al modificar un documento compartido.

```
11. #ProcesoDeFirma.anular() {  
  this.estaActivo = false;  
  this.colaboradores.forEach(it => it.anular())  
}
```

podrían  
renovarlos

esto es antes que se  
libere entonces esta  
deshabilitado

```
# Colaboracion.anular() {  
  this.estado = DESHABILITADA  
}
```

```
# RepositorioProcesos.getAll() {  
  return this.procesos.filter(it => it.estaActivo())  
}
```

donde se usa? como?

```
# ProcesoDeFirma.esCobrable(mes) {  
  return (estaLibrado || estaActivo) && fechaGeneracion.completo(mes) == 0  
}
```

se compara entre fechas  
usando la API del lenguaje