<https://medium.com/karibu-blog/c%C3%B3mo-enviar-un-mail-con-spring-boot-f86c2f7af678>

# ¿Cómo enviar un mail con Spring Boot?

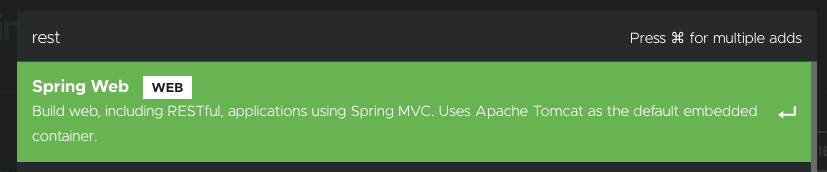
Corta implementación de un proyecto para enviar mails.

Según Wikipedia: un correo electrónico es un [servicio de red](https://es.wikipedia.org/wiki/Servicio_de_red) que permite a los usuarios enviar y recibir mensajes (también denominados [mensajes](https://es.wikipedia.org/wiki/Mensaje) electrónicos o cartas digitales) mediante [redes de comunicación electrónica](https://es.wikipedia.org/wiki/Red_de_telecomunicaci%C3%B3n).

Los sistemas de correo electrónico se basan en un modelo de [almacenamiento y reenvío](https://es.wikipedia.org/wiki/Almacenamiento_y_reenv%C3%ADo), de modo que no es necesario que ambos extremos se encuentren conectados simultáneamente. Para ello se emplea un servidor de correo que hace las funciones de intermediario, guardando temporalmente los mensajes antes de enviarse a sus destinatarios.

En este caso, se implementará un pequeño servicio aprovechando la herramienta que Spring provee para el envío de correo el cual es una herramienta para para usar el soporte de envío de correo electrónico de Java Mail y Spring Framework, al final del post se dejara un proyecto de ejemplo. Donde se implementa todo lo que aquí se explicará.

Para comenzar se puede acudir a <https://start.spring.io/> para generar un proyecto base en el cual se deberá agregar la dependencia:



Luego de esto podemos dar click en generar para descargar el formato zip nuestro proyecto.

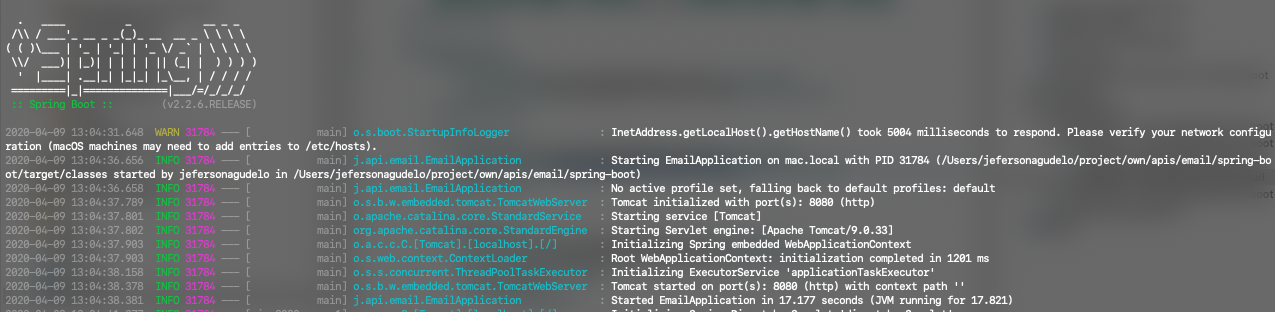
Una vez descargado y descomprimido el proyecto se podrá iniciar el proyecto. Esto se puede hacer con maven(en caso de que se tenga instalado) con el comando:

mvn spring-boot:run

o bien, con el archivo mvnw que esta en la carpeta descargada del proyecto.

./mvnw spring-boot:run

Esto levantará la aplicación basica.



Luego de esto será necesario agregar la siguiente dependencia para poder tener acceso a las clases necesarias para enviar el mail, esto en nuestro archivo pom.xml:

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-mail</artifactId>  
</dependency>

Posterior a esto se deben agregar las siguientes clases:

server.port=8080  
spring.mail.host=smtp.gmail.com  
spring.mail.port=587  
spring.mail.username={{email}} #email para enviar  
spring.mail.password={{contraseña}} #contraseña de aplicación  
spring.mail.properties.mail.smtp.starttls.enable=true  
spring.mail.properties.mail.smtp.starttls.required=true  
spring.mail.properties.mail.smtp.auth=true  
spring.mail.properties.mail.smtp.connectiontimeout=5000  
spring.mail.properties.mail.smtp.timeout=5000  
spring.mail.properties.mail.smtp.writetimeout=5000

Donde:

* **email:** Sera el emisor del mail.
* **contraseña:** NO es la contraseña del email, es una contraseña especial de aplicación que provee gmail. Se obtiene de la siguiente manera:

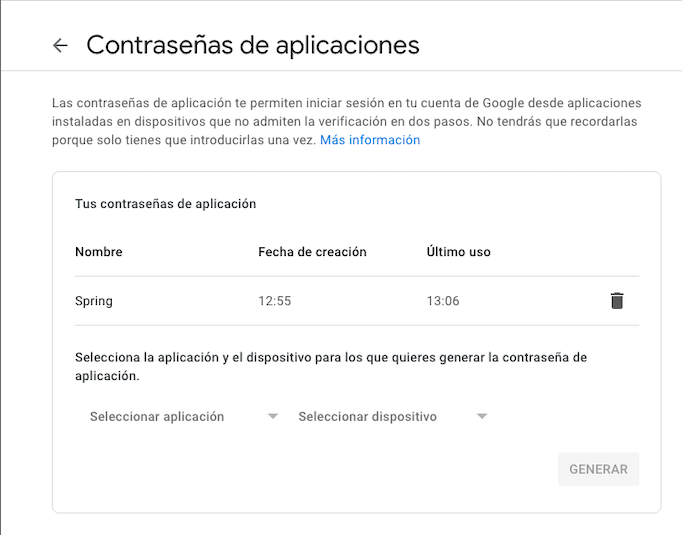
Nota: Si utilizas la [verificación en dos pasos](https://support.google.com/accounts/answer/185839) y ves el error “Contraseña incorrecta” al intentar acceder a tu cuenta de Google, es posible que puedas solucionarlo usando una contraseña de aplicación.

1. Ve a tu [cuenta de Google](https://myaccount.google.com/).
2. En el panel de navegación de la izquierda, selecciona Seguridad.
3. En el panel “Iniciar sesión en Google”, haz clic en Contraseñas de aplicaciones. Si no aparece esta opción, puede ser por estos motivos:

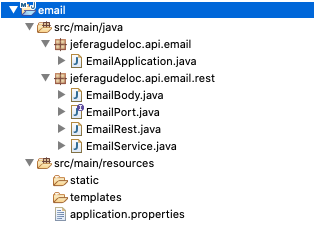
* La verificación en dos pasos todavía no está configurada en la cuenta.
* La verificación en dos pasos solo está configurada para llaves de seguridad.
* La cuenta pertenece a tu trabajo, centro educativo u otra organización.
* Has activado la función Protección Avanzada en la cuenta.

1. En la parte inferior, haz clic en Seleccionar aplicación y elige la aplicación que estés utilizando.
2. Haz clic en Seleccionar dispositivo y elige el dispositivo que estés utilizando.
3. Selecciona Generar.
4. Sigue las instrucciones para introducir la contraseña de aplicación. Esta contraseña es el código de 16 caracteres que aparece en una barra amarilla, en tu dispositivo.
5. Selecciona Listo.

La mayoría de las veces, solo tendrás que introducir una contraseña de aplicación una vez por aplicación o dispositivo, de modo que no es necesario que la memorices.



Luego de este paso, bastante importante se procederá a crear lo siguiente:



Estructura basica.

Donde EmailRest será nuestro controlador que estará escuchando las llamadas desde afuera, EmailBody será el DTO de entrada para mandar el email, el EmailPort será la interfaz para que el EmailRest utilice el EmailService que será finalmente nuestra clase que tenga la lógica para enviar el mail.

Lógica de envio de mail.

Finalmente para que funcione este Mail Sender se debe llamar a

host:puerto/email/send

En caso de que se levante en el local:

<http://localhost:8080/send/email>

Es un método post, donde el body debe lucir de la siguiente manera:

*{  
“email”:”*[*ejemplo@gmail.com*](mailto:agudelocontrerasj@gmail.com)*”,  
“content”:”Contenido de email”,  
“subject”:”Subject de email”  
}*

Y eso es todo, debería entregar un true en caso de ser enviado o false en caso de tener problemas.