

Dentro de namecheap tendremos una opción de agregar a nuestro dominio un servicio de correos, este será fácil de configurar y de usar dentro de nuestras aplicaciones web.



Dashboard



Expiring / Expired



Domain List



Hosting List



Private Email



SSL Certificates



Apps



My Offers

NEW



Profile

Cuándo nosotros estemos dentro de **NameCheap** veremos que tendremos de lado del lateral izquierdo una serie de opciones con los cuáles podemos personalizar o configurar nuestro dominio.

En el apartado que nos enfocaremos será el de Private Email:

### Private Email Subscriptions

Subscription	Plan	Status	Auto-Renew	Expiration	
thinkguille.space	Starter	✓ ACTIVE	<input checked="" type="checkbox"/>	Sep 6, 2025	MANAGE

Esto será el lugar dónde nosotros podremos ver nuestras suscripciones activas, en caso de que seamos nuevos usando este servicio veremos que nos ofrece un plan de prueba gratuito, una vez que nosotros "compremos" este plan veremos que la interfaz cambia

Private Email – Starter Email Subscription TRIAL ✓ ACTIVE  
for **thinkguille.space**

Upgrade plan

You're almost set! To be able to activate your Private Email subscription to receive mail and create mailboxes, you must first set up these important DNS records from the table below. You can find a little help to do this in our handy [step-by-step guide](#). Once completed, please allow up to **4 hours** for the changes to take effect.

Hostname	Record type	Priority	Value
@	MX	10	mx1.privateemail.com
@	MX	10	mx2.privateemail.com
@	TXT		v=spf1 include:spf.privateemail.com ~all

Validity

Jul 6, 2025 - Sep 6, 2025 Cancel

RENEW

☒ Auto-renew every year

Once the trial period is over, your subscription will be renewed for the following year.

Assigned Email Storage



5 GB of 5 GB assigned to mailbox

File Storage



0 Bytes of 200 MB in use

Aquí nos dirá nuestro plan y nos dirá que es importante tener tanto un mailbox configurado, cómo agregar a nuestro entregador de dominio nuestra configuración esencial para usar el protocolo de correos.

## Cloudflare

En mi caso manejé cloudflare, por lo que de manera automática este servicio jaló la configuración básica de mi suscripción y la indexó dentro de mi zona (entorno), por lo que lo único que debí meter de manera manual es una llave de dominio y una llave DKIM, esta es accesible si nosotros vamos a la parte inferior de nuestro plan de correos:

## DKIM

- Make a copy of your DKIM record values below. Keep this handy.
- Visit your DNS provider and set up your DKIM as a TXT record.

### DNS Record

default.\_domainkey IN TXT ("v=DKIM1;k=rsa;p=MIIBljANBgkq

COPY

### Public Key

-----BEGIN PUBLIC KEY-----MIIBljANBgkqhkiG9w0BAQEFA

COPY

### Private Key

-----BEGIN RSA PRIVATE KEY-----MIIEpAIBAAKCAQEA1LW

COPY

HIDE DKIM

REMOVE DKIM

Your unique DKIM keys will sign every email message you send as from a trusted source. You can choose to 'SHOW' or 'HIDE' your view of the keys at any time.

### Anti-Spoof Filter



Use this filter to detect and stop harmful spoof attacks. [Learn more →](#)

Esa llave la pondremos como archivos TXT dentro de nuestro perfil de Cloudflare

<input type="checkbox"/>	Tipo ①	▲	Nombre ①	Contenido ①	Estado de proxy ①	TTL ①	Acciones
<input type="checkbox"/>	A		thinkguille.space	161.35.56.53	Solo DNS	Automático	<a href="#">Editar ▶</a>
<input type="checkbox"/>	A		www	161.35.56.53	Solo DNS	Automático	<a href="#">Editar ▶</a>
<input type="checkbox"/>	TXT		default._domainkey	"v=DKIM1; k=rsa; p=MII...	Solo DNS	Automático	<a href="#">Editar ▶</a>
<input type="checkbox"/>	TXT		_dmarc	"v=DMARC1; p=quaranti...	Solo DNS	Automático	<a href="#">Editar ▶</a>

Bastará con poner estas dos llaves, para que reconozca el tipo MX para los correos.

# Servicio de MailBox

Al momento que nosotros escogemos nuestro paquete va a pedir que registremos el correo y la contraseña de nuestra cuenta, esa por default será "NoReplay@nuestrodominio", una vez que terminemos de configurar esta cuenta podremos acceder a ella.

1 of 1 Mailboxes In Use BUY MAILBOX

Mailbox	Status	Aliases	Email Storage
support@thinkguille.space	ON	0 of 10	0.0% Used 5 GB Available

EDIT STORAGE

Done

Podemos tener la cantidad que nosotros deseemos, siempre y cuándo paguemos por ellas, así como si queremos que tengan más o menos capacidad de disco para los mensajes, o que habilitemos la opción de envío desde el servicio, irán alterando el precio que pagaremos por este mismo.

Liga de acceso: <https://privateemail.com>

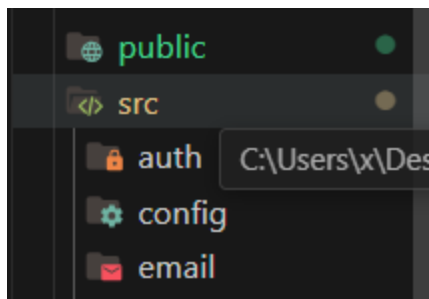
Aquí pondremos las credenciales que configuramos al momento de crear nuestro servicio de correo.

Una vez que accedemos podremos ver todos nuestros correos.

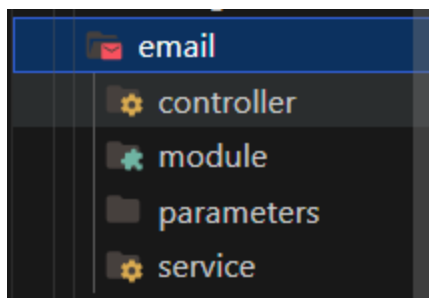
The screenshot shows the PrivateEmail web interface. The top navigation bar includes the PrivateEmail logo, a search bar, and various icons for settings, help, and user profile. Below the navigation bar, there are tabs for different email categories: General (3), Promotions, Social, Purchases, Friends, and Work. The main content area is divided into two sections. On the left, there's a sidebar with a list of folders: Inbox (3), Drafts, Sent, Spam, and Trash. Below the folders, there's a link to 'thinkguille.space' and a 'Mail quota' section showing '28 KB of 5 GB'. The main email list shows three emails: 'Lead Example' (Yesterday), 'Ejemplo ahora' (Yesterday), and 'Eduardo' (9:12 PM). The 'Lead Example' email is selected, and its details are shown on the right. The email is titled 'Nuevo mensaje desde la aplicación' and is from 'Lead Example' to 'support@thinkguille.space'. The body of the email contains contact information: 'Nombre: Lead Example', 'Correo: direc@gmail.com', and 'Teléfono: 83272'. The email is marked as a lead with the text 'Este es un lead'.

## Configurando el back

Para consumir este servicio dentro de nuestra aplicación podemos manejarlo de la manera directa o estructurada, al ser un servicio de consumo externo, nosotros podemos jugar un poco con la manera en que hacemos que este cargue en nuestro servidor.



Yo decidí ponerlo dentro de la carpeta "email", no es obligatorio, pero es parte de un estándar el dividir estas secciones según su funcionalidad.



Aquí entra la parte de la adaptabilidad a nuestras necesidades, ya que podemos trabajar el modelado en capas o trabajar toda la lógica en nuestro controlador.

---

## Herramientas adicionales

He decidido que podemos aprovechar la conexión STP que nos genera por defecto el servicio (cuentas institucionales), para aprovechar las ventajas que nos ofrece **NodeMailer**

Con esta dependencia nosotros solamente deberemos crear nuestro constructor y generar así un entorno dirigido.

```

constructor() {
  this.transporter = nodemailer.createTransport({
    host: 'mail.privateemail.com',
    port: 587,
    secure: false,
    auth: {
      user: 'support@thinkguille.space',
      pass: 'Alisson050125#',
    },
    tls: {
      rejectUnauthorized: false,
    },
  });
}

```

Aquí es importante que prestemos atención a la documentación de esta librería, ya que depende de nuestro servicio será el puerto que deberemos activar y si ponemos o no el modo seguro

Link de la documentación: <https://www.npmjs.com/package/nodemailer>

---

## Template.ts

```
import { CreateFormDto } from "src/formulario/dto/formulario.dto";
```

```
export function createEmailParameters(data: CreateFormDto) {
```

```
    const { nombre_completo, correo, telefono, mensaje } = data;
```

```
    const plainText = `
```

```
    Solicitud de contacto desde la aplicación:
```

```
    Nombre: ${nombre_completo}
```

Correo: \${correo}

Teléfono: \${telefono}

Mensaje:

\${mensaje}

```
` .trim();
```

```
const html = `
```

```
<html>
```

```
<body style="font-family: Arial, sans-serif; color: #333;">
```

```
<h2>Solicitud de contacto desde la aplicación</h2>
```

```
<p><strong>Nombre:</strong> ${nombre_completo}</p>
```

```
<p><strong>Correo:</strong> ${correo}</p>
```

```
<p><strong>Teléfono:</strong> ${telefono}</p>
```

```
<p><strong>Mensaje:</strong></p>
```

```
<blockquote style="background:#f9f9f9; padding:10px; border-  
left:5px solid #ccc;">
```

```
    ${mensaje}
```

```
</blockquote>
```

```
</body>
```



```
        </html>

        `;

    return {

        email: 'support@thinkguille.space',

        name: nombre_completo,

        message: plainText,

        html: html,

        title: 'Nuevo mensaje desde la aplicación',

    };

}
```

En este apartado, solamente vamos a manejar la estructura de nuestro correo, cómo queremos pintar este en nuestras notificaciones y el email al que irá dirigido, este será el que nosotros configuramos en nuestra caja de correos.

---

## Protección

Aunque en nuestro constructor pudimos ver los parametros, por estándar de seguridad es recomendable que migremos esa configuración al .env en caso de que queramos deployar en producción:

```
SMTP_HOST=mai
SMTP_PORT=587
SMTP_SECURE=f
SMTP_USER=sup
SMTP_PASS=Ali
```

---

# Configuraciones adicionales

En mi caso mi servidor es ssh, por lo que si queremos que nuestras peticiones salgan de manera correcta, debemos especificar en nuestro archivo

## Email.Module.ts

```
You, 3 days ago | 1 author (You)
import { Module } from "@nestjs/common";
import e from "express";
import { EmailJsService } from "../service/emailjs.service";
import { ConfigModule } from "@nestjs/config";
import { HttpModule } from "@nestjs/axios";

You, 3 days ago | 1 author (You)
@Module({
  imports: [ConfigModule, HttpModule],
  providers: [EmailJsService],
  exports: [EmailJsService],
})
export class EmailModule {}
```

You, 3 days ago • FEAT: ADD SLACK

Que esperamos una configuración HTTP y funciones Axios adicionales

---

## Imports

Vamos a declarar tanto módulos como servicios en la configuración de

## App.module.ts

```

@Module({
  imports: [
    ConfigModule.forRoot({
      isGlobal: true,
      load: [recaptchaConfig, slackConfig, emailConfig],
    }),
    TypeOrmModule.forRoot({
      type: 'sqlite',
      database: process.env.DATABASE_PATH || './db.sqlite',
      entities: [User, Lead],
      synchronize: true, // Solo en desarrollo
    }),
    FormularioModule,
    RecaptchaModule,
    AuthModule,
    EmailModule
  ],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule {}

```

De esta manera haremos que se pueda acceder manera global a todas las configuraciones que nosotros hayamos definido.

## Uso

Por último bastará con que declaremos nuestra variable y definamos en que momento esperamos que se dispare nuestro servicio para mandar nuestro correo de la app a la caja de correos.

## Formulario.controller.ts

```

await this.slackService.sendMessageToSlack(data);
await this.emailService.sendEmail(data);

```