DOCUMENTACIÓN Y MANUAL

PRUEBA TÉCNICA PARA PUESTO DESARROLLADOR FULL STACK

DESARROLLADO POR: KEVIN DANIEL SIERRA CASTRO

ESTUDIANTE DE INGENIERÍA DE SISTEMAS EN LA CORPORACIÓN UNIVERSITARIA RAFAEL NÚÑEZ

FECHA DE CREACIÓN: 13/04/2023

DESARROLLO DE UNA APLICACIÓN WEB QUE PERMITA GESTIONAR INFORMACIÓN DE EMPLEADOS Y ENVIAR CORREOS AUTOMÁTICOS

INTRODUCCIÓN

La presente aplicación fue hecha con **Django,** un framework de desarrollo web de código abierto basado en Python. **React**, una biblioteca de JavaScript de código abierto para diseñar interfaces de usuarios interactivas y dinámicas. **PostgreSQL,** un sistema de gestión de bases de datos relacional orientado a objetos y de código abierto. Además se utilizó otras dependencias para implementar un mejor funcionamiento de la aplicación, estas serán explicadas con detalles más adelante.

La aplicación consiste en gestionar información sobre empleados como también automatizar el envío de correos electrónicos de bienvenida a los mismos, esto es gracias mediante una interfaz donde hay un formulario que el usuario llena con sus datos y estos son enviados a la API REST desarrollada con el framework **djangorestframework**, una vez almacenado los datos en la API el usuario podrá hacer acciones como modificar y eliminar sus datos en la misma interfaz incluyendo mensajes de correos de acuerdo a sus acciones, los datos que el usuario envíe, modifique y elimine, también son actualizados en la base de datos.

INSTALACIÓN Y CONFIGURACIÓN DE LA APLICACIÓN

Backend + Base de datos + Servidor de correos

Para este entorno se necesita tener instalado Python, Django y las siguientes dependencias:

Package	Version
asgiref	3.8.1
certifi	2024.2.2
charset-normalizer	3.3.2
coreapi	2.3.3
coreschema	0.0.4
distlib	0.3.8
Django	5.0.4
django-cors-headers	4.3.1
djangorestframework	3.15.1
djangorestframework-simplejwt	5.3.1
filelock	3.13.4
idna	3.7
itypes	1.2.0
Jinja2	3.1.3
MarkupSafe	2.1.5
platformdirs	4.2.0
psycopg2	2.9.9
PyJWT	2.8.0
requests	2.31.0
resend	0.8.0
sqlparse	0.4.4
tzdata	2024.1
uritemplate	4.1.1
urllib3	2.2.1
virtualenv	20.25.1

Para ver qué dependencias o packages tienes, ejecuta este comando en la consola dentro del app backend: **pip list,** verás la lista de dependencias, si te falta alguna por favor instalarla.

Configuración y explicación en backend/backend

en el archivo settings.py necesitas configurar lo siguiente:

```
ALLOWED_HOSTS = ['127.0.0.1', 'localhost']
```

Es el anfitrión donde se ejecuta la API REST

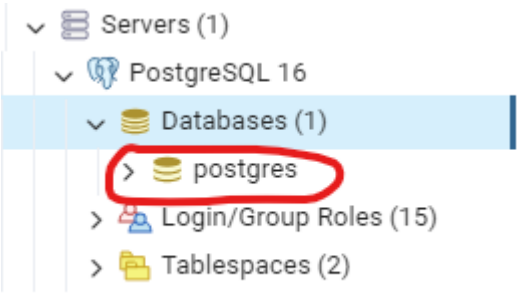
```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'corsheaders',
    'rest_framework',
    'coreapi',
    'employee'
]
```

Estas aplicaciones proporcionan funcionalidades esenciales para desarrollar una aplicación web con Django, como autenticación de usuarios, administración de contenido, gestión de sesiones, manejo de solicitudes de API, entre otros.

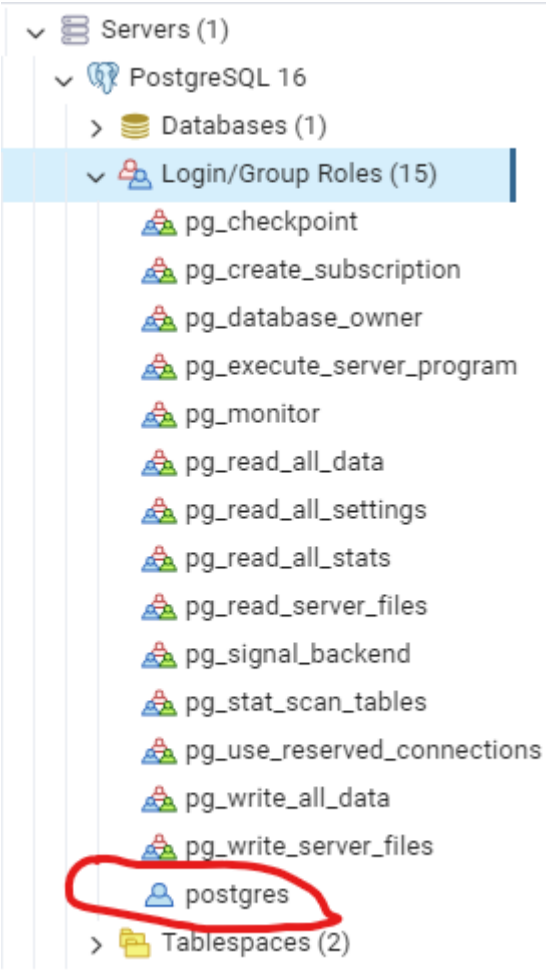
```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': "postgres",
        'USER': 'postgres',
        'PASSWORD': 'pruebatecnica',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

Es la configuración automática de la base de datos, debes tener instalado PostgreSQL en tu ordenador y una vez instalado y configurado el **name**(nombre de tu base de dato), **user**(es por defecto), **password**(la que pusiste durante la instalación), **host**(es por defecto), **port**(es por defecto), lo configuraras en **DATABASES**

Ejemplo:
-Nombre de tu base de datos



-Nombre de tu usuario




```
CORS_ALLOWED_ORIGINS = ["http://localhost:5173"]
```

Este es el dominio donde la aplicación React se está ejecutado, si tu aplicación está en el puerto 3000 debería ser así: `CORS_ALLOWED_ORIGINS = ["http://localhost:3000"]`

```
EMAIL_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587  # El puerto utilizado por tu servidor SMTP (por
ejemplo, 587 para TLS)
EMAIL_USE_TLS = True  # Indica si se debe usar TLS para cifrar la
conexión
EMAIL_HOST_USER = 'kevnsc18@gmail.com'
EMAIL_HOST_PASSWORD = 'qfko bzfl hnvb dtot'
```

Es la configuración para el envío de correos, en este caso con gmail

EMAIL_HOST_USER debe ser tu email con el cual estás registrado en gmail

EMAIL_HOS_PASSWORD se configura desde la administración de tu cuenta de google, luego vas a seguridad y asegurate de tener activada la verificación de dos pasos, en la parte superior hay un buscador, escribe contraseñas de aplicaciones y crea una, esa es la que debes poner en `EMAIL_HOST_PASSWORD`

En urls.py incluye estas rutas

```
from django.contrib import admin
from django.urls import include, path

urlpatterns = [
    path('admin/', admin.site.urls),
    path('employee/', include('employee.urls'))
]
```

Son para ir a la pagina de administracion y ver tu API REST

Configuración en backend/employee

En admin.py registra el model Employee

```
from django.contrib import admin
from .models import Employee

# Register your models here.
admin.site.register(Employee)
```

En models.py veras la configuración del modelo Employee el cual tendrá los campos necesarios donde el usuario va a enviar sus datos y la lógica necesaria para enviar los correos automatizados usando signals:

```
from django.db.models.signals import post_save, post_delete
from django.dispatch import receiver
from django.core.mail import send_mail
from django.conf import settings
```

En serializer.py se convierte las instancias de Employee en representaciones de datos compatibles con JSON, es la construcción de la API REST para manipular objetos Employee.

En urls.py se define la ruta de la api

```
from django.urls import include, path
from rest_framework.documentation import include_docs_urls
from rest_framework import routers
from employee import views

router = routers.DefaultRouter()
router.register(r'employee', views.EmployeeView, 'Employee')

urlpatterns = [
    path("api/v1/", include(router.urls)),
    path('docs/', include_docs_urls(title="Employee API"))
]
```

En views.py se define una vista de Django REST Framework que proporciona endpoints RESTful para interactuar con el modelo Employee

Ya configurado todo faltaría hacer las migraciones para guardar todos los cambios:

primero para la app employee (no se necesitar ir a backend/employee, simplemente desde backend)

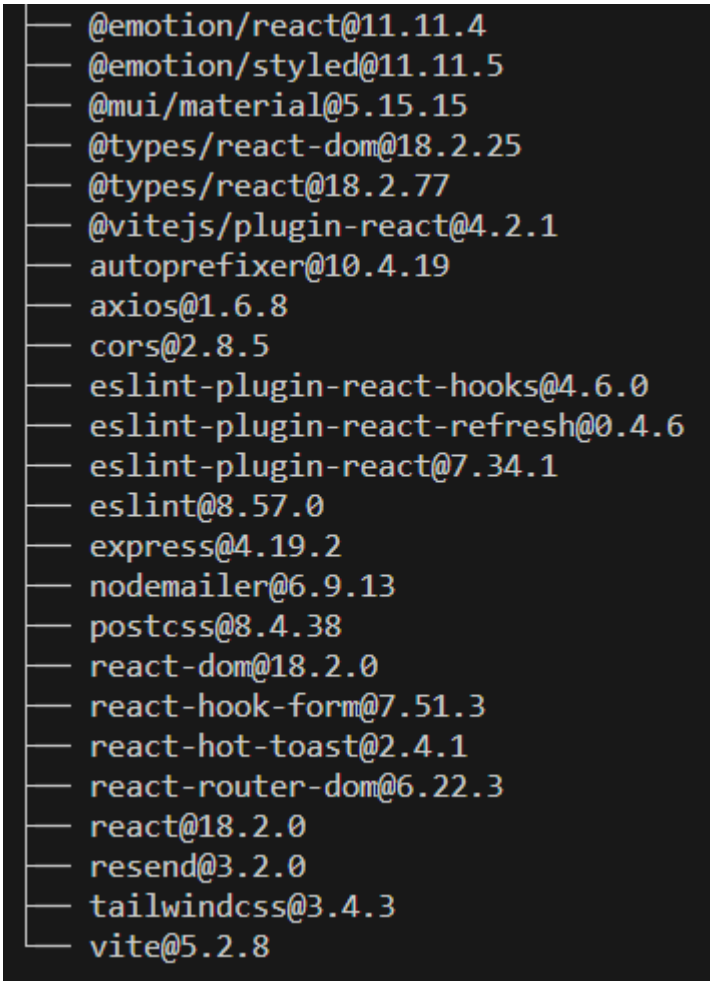
python manage.py makemigrations employee
python manage.py migrate

segundo para todo en general

python manage.py makemigrations employee
python manage.py migrate

Frontend

Para este entorno se utilizó Vite + React, material UI para crear la aplicación, junto con otras dependencias las cuales debes instalar:



Dicho anteriormente verifica si tienes las siguientes dependencias con **npm list** e instalar si te falta alguna

- Explicación y configuración

En src/App.jsx se renderizan las rutas para los componentes y Toaster para las notificaciones cuando el usuario haga un acción

En src/Api/employee.api.js es el endpoint de la API REST que contiene los métodos como get, post, delete y put

En src/Components/EmployeeForm.jsx es un componente donde se crea la interfaz del formulario usando los componentes de la liberia `@mui/material.`
En este componente se llama los endpoints de employee.api.js para hacer las funciones específicas dentro del componente como crear, modificar y eliminar

En src/Components/EmployeeList.jsx es un componente donde se lista los usuarios(empleados) con los los componentes de la liberia `@mui/material`.
En este componente se llama un endpoint el cual obtiene todos los empleados y los renderiza si existen, en caso de que no aparece un mensaje de que no existen

En src/Navigation/Navigation.jsx es un componente para ir a las rutas donde se lista los empleados o ir al formulario

Si ya está instalado tailwind asegurate de que en el archivo tailwind.config.js esté exportado las extensiones y carpetas

```
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

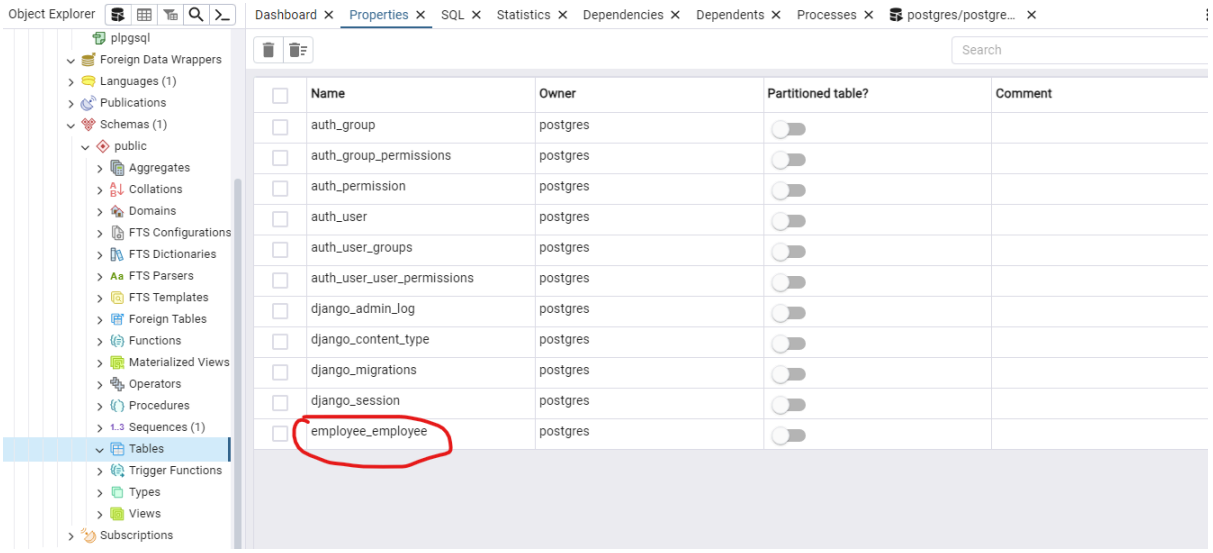
Base de datos

Para ver qué datos se han enviado a la base de datos ves a Databases>postgres (la base de datos por defecto, si tus datos se envían a otra base de datos dirígete a ella) > Schemas>Tables

Luego haz click derecho en Tables, seleccionas Query Tools y escribe y ejecuta este comando

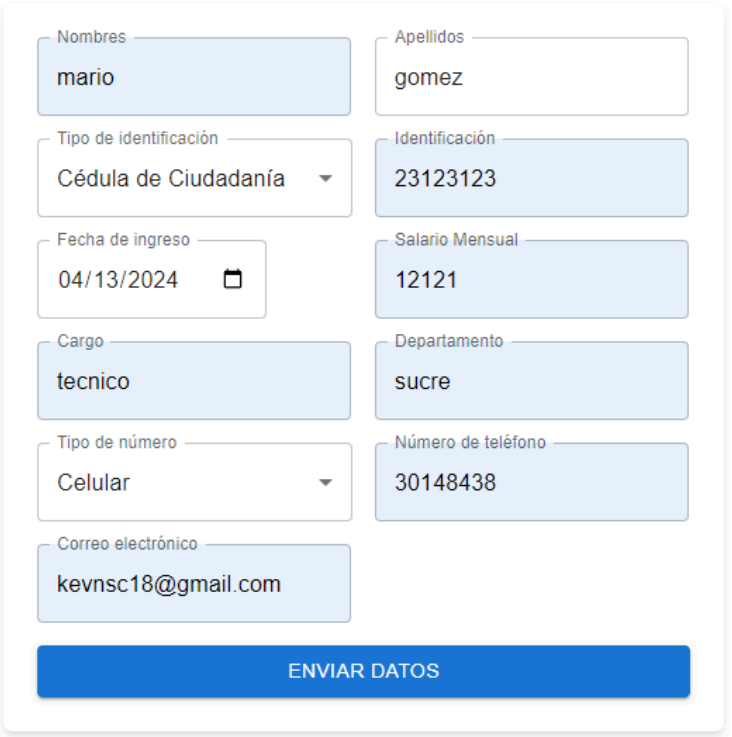
SELECT * FROM employee_employee

es la tabla donde se guardan las propiedades, si la tuya no es así reemplazar por el nombre de tu tabla

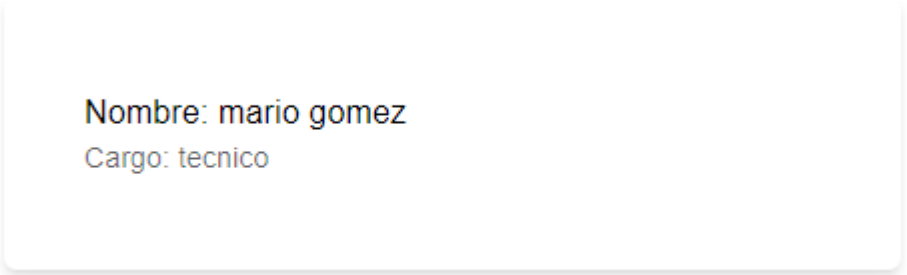


Interfaz frontend

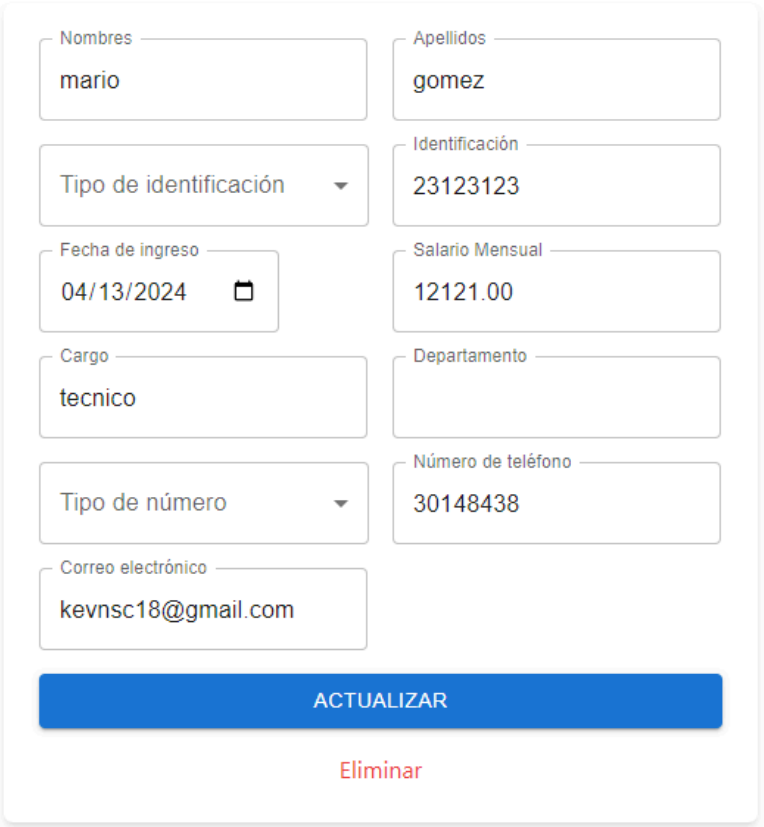
-Formulario



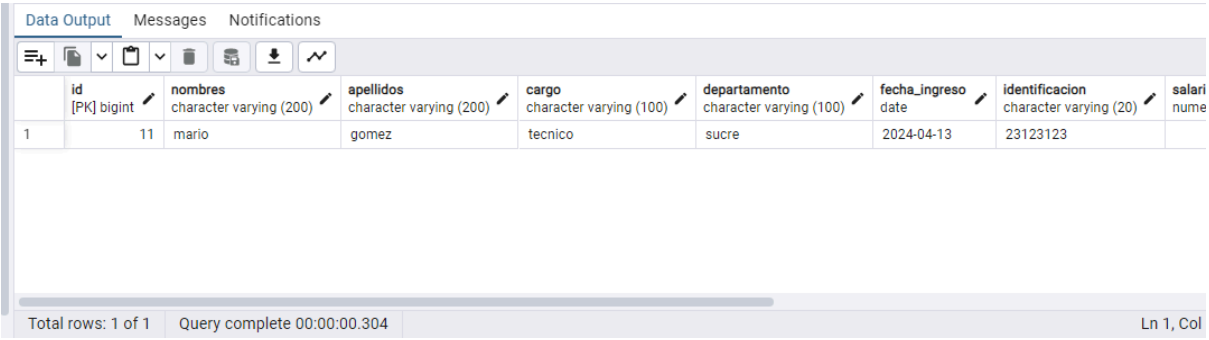
-Lista



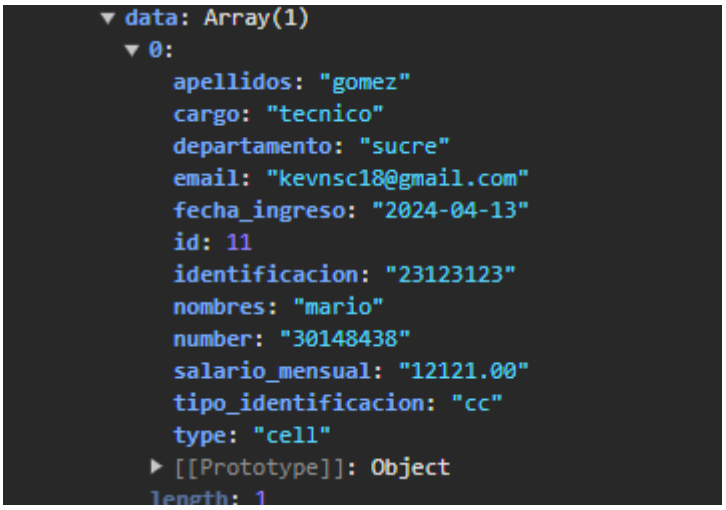
-Modificar y eliminar datos



Base de datos



Datos de la Api



Repositorios

https://github.com/KevinDaniel18/PRUEBA-TECNICA-FRONTEND
https://github.com/KevinDaniel18/PRUEBA-TECNICA-BACKEND