

Proyecto REST

Estado climático en chile

Axel Brito González – Cristóbal González Gárate – Francisco Díaz Hernández

| Sebastián Salazar Molina | Computación Paralela y Distribuida | 30-10-2021

Contenido

[Índice de ilustraciones 2](#_Toc86529506)

[1.- Introducción 2](#_Toc86529507)

[2.- Resolución del problema 2](#_Toc86529508)

[2.1.- Extracción de información 3](#_Toc86529509)

[2.1.1.- Primera fuente de información 3](#_Toc86529510)

[2.1.2.- Segunda fuente de información 3](#_Toc86529511)

[2.1.3.- Tercera fuente de información 4](#_Toc86529512)

[2.2.- Implementación de información 5](#_Toc86529513)

[2.3.- Mecanismo de autenticación 5](#_Toc86529514)

[2.4.- Implementación de la api 6](#_Toc86529515)

[2.4.1.- Log in 6](#_Toc86529516)

[2.4.2.- Estaciones 6](#_Toc86529517)

[2.4.3.- Búsqueda 6](#_Toc86529518)

[2.5.- Funcionamiento general de la api 6](#_Toc86529519)

[2.5.1.- Extracción de Información 7](#_Toc86529520)

[2.5.2.- Consulta de datos 7](#_Toc86529521)

[2.6.- Funcionamiento de la API 7](#_Toc86529522)

[3.- Tecnología utilizada 8](#_Toc86529523)

[3.1.- Lenguaje utilizado 8](#_Toc86529524)

[3.2.- Máquina Virtual ORACLE VM Virtualbox 8](#_Toc86529525)

[3.3.- Dependencias de NodeJS 8](#_Toc86529526)

[3.3.1.- Cheerio 8](#_Toc86529527)

[3.3.2.- Bcryptjs 8](#_Toc86529528)

[3.3.3.- CORS 8](#_Toc86529529)

[3.3.4.- DotenV 8](#_Toc86529530)

[3.3.5.- Express 8](#_Toc86529531)

[3.3.6.- JsonWebToken 9](#_Toc86529532)

[3.3.7.- Morgan 9](#_Toc86529533)

[3.3.8.- Node 9](#_Toc86529534)

[3.3.9.- PG 9](#_Toc86529535)

[3.3.10.- Nodemon 9](#_Toc86529536)

[3.4.- Uso de PostgreSQL 9](#_Toc86529537)

[3.5.- Uso de Github 9](#_Toc86529538)

[3.6.- Uso de Visual Studio Code 10](#_Toc86529539)

[3.7.- Uso de Postman 10](#_Toc86529540)

[4.- Conclusión 10](#_Toc86529541)

# Índice de ilustraciones

[Ilustración 1: Primera fuente de información 3](#_tyjcwt)

[Ilustración 2: Segunda fuente de información 3](#_1t3h5sf)

[Ilustración 3: Tercera fuente de información 4](#_2s8eyo1)

[Ilustración 4: Traspaso de información a Base de datos 4](#_3rdcrjn)

[Ilustración 5: Base de datos de la API 5](#_26in1rg)

[Ilustración 6: Funcionamiento JWT 6](#_35nkun2)

[Ilustración 7: Funcionamiento API REST 6](#_44sinio)

[Ilustración 8:Funcionamiento general de la API 7](#_4i7ojhp)

# 1.- Introducción

El cambio climático es una realidad cada vez más apremiante. Así lo asegura toda la evidencia científica generada durante los últimos años, por lo que la humanidad ha adoptado una serie de medidas y acuerdos para enfrentar este fenómeno global. El aumento de las temperaturas, la cada vez mayor proliferación de fenómenos meteorológicos extremos, inundaciones y sequías son sólo algunas de las consecuencias que ya se sienten debido al cambio climático, siendo nuestro país particularmente vulnerable a ellas.

No es desconocido para nadie que nuestro país atraviesa una de las mayores mega sequías de la última década y un sostenido aumento de las temperaturas. La disminución de las lluvias tiene enormes consecuencias en la generación de electricidad, nuestra agricultura y la vida diaria, la falta de agua, se nota.

Por este motivo, se desarrollará un sistema que permita disponer el histórico de estos datos, a través de un API REST simple de consumir.

# 2.- Resolución del problema

Como ya se ha dicho anteriormente, en este proyecto se realizará una API REST que tenga el histórico de los datos con respecto al estado climático de Chile.

Se obtendrá la información de distintas fuentes:

* El departamento de Geofísica de la Universidad de Chile en conjunto con el departamento de Geofísica de la Universidad de Concepción, otorgando información desde los años 1950 hasta el año 2005. Los indicadores de estos datos serían: Nombre de la estación, coordenadas, período, precipitación, temperatura máxima y temperatura mínima.
* Luego el gobierno de Chile impulsó una campaña de apertura de información que ha permitido complementar la fuente anterior, con datos desde el 2013 hasta el 2021. Los indicadores que otorga serían: Código, nombre de la estación, coordenadas, altura con respecto al mar, año, mes, día, precipitación, temperatura mínima y temperatura máxima.
* Por último, desde la página de la Dirección Meteorológica de Chile – Servicios Climáticos, la cual es información que se va actualizando día a día. Los indicadores que entrega son: Fecha, hora, temperatura mínima, temperatura máxima y precipitaciones.

*Teniendo las fuentes, se procede a analizar cómo trabajar los datos obtenidos.*

## 2.1.- Extracción de información

Para extraer la información de distintas fuentes, requiere tener conocimiento para poder realizar web scraping y también extracción de datos de fuentes estáticas.

### 2.1.1.- Primera fuente de información

Para trabajar la primera fuente de información, necesitamos dividir los indicadores

*Ilustración 1: Primera fuente de información*

Imagen que contiene Texto

Descripción generada automáticamente

Como es una fuente estática (archivo Excel), se procede a leer la información de ésta y transformarla de manera que se pueda guardar en una base de datos.

### 2.1.2.- Segunda fuente de información

La segunda fuente de información es algo más compleja de obtener información, pero en simples palabras, se trabajaría de la misma manera que la primera fuente de información.

*Ilustración 2: Segunda fuente de información*

Carta

Descripción generada automáticamente con confianza media

La manera como está distribuida la información de esta fuente, son diversas subpáginas de la misma página que posee una gran cantidad de archivos Excel.

### 2.1.3.- Tercera fuente de información

Para la tercera fuente de información, ésta se actualiza constantemente entonces la única manera que se tiene que trabajar sería con web scraping.

*Ilustración 3: Tercera fuente de información*

Escala de tiempo

Descripción generada automáticamente

La página de la Dirección Meteorológica de Chile – Servicios Climáticos, tiene bastante información, pero sólo usaremos el nombre de la estación, T° Máxima, Precipitación, T° Mínima y hora.

Comparado a las fuentes estáticas que fueron mostradas anteriormente, de acá se tendrá que recopilar información constantemente. El web scraping se tendría que automatizar para poder obtener información cada cierto tiempo, esta información extraída tendría que ser pasada a una base de datos para poder ser mostrada más adelante.

## 2.2.- Implementación de información

Luego de extraer exitosamente la información, ésta se debe procesar para ser guardada dentro de una base de datos, de manera que pueda ser mostrada por la API.

*Ilustración 4: Traspaso de información a Base de datos*

Un dibujo animado con letras

Descripción generada automáticamente con confianza baja

Se crearán las tablas con la información requerida.

*Ilustración 5: Base de datos de la API*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Teniendo la información dentro de la base de datos, es más sencillo de buscar los datos y trabajarlos si es necesario. Dentro de la ilustración 5, se puede divisar una tabla de usuario que se explicará más adelante al respecto de éste.

## 2.3.- Mecanismo de autenticación

Se trabajará con JSON Web Token (jwt), el cuál es un estándar abierto basado en JSON. De manera que pueda generar tokens para proteger la API al momento de ser consumida.

*Ilustración 6: Funcionamiento JWT*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

## 2.4.- Implementación de la api

Teniendo el mecanismo de autenticación ya visto, se puede comenzar con la API REST, de manera que se pueda realizar una aplicación web con las buenas prácticas utilizadas en las requisiciones HTTP.

*Ilustración 7: Funcionamiento API REST*



Las operaciones que se realizarán serán las siguientes:

### 2.4.1.- Log in

Con la seguridad mostrada anteriormente, esta operación recibirá la información necesaria para autenticar un consumidor válido del API. En esta ocasión, se utilizará correo y contraseña.

### 2.4.2.- Estaciones

Estaciones: Esta operación mostrará el nombre de las estaciones con los datos que se encuentren en la base de datos. Ya que se explicó la información que está dentro de la BDD, se puede predecir que podría retornar con esta operación.

### 2.4.3.- Búsqueda

La siguiente operación buscaría la información según los indicadores (como nombre, fecha, etc) y retornaría un grupo de información que corresponda al indicador que se buscó. Haciendo consultas a la BDD ya poblada, se puede hacer con más facilidad.

## 2.5.- Funcionamiento general de la api

En resumidas cuentas, el aplicativo debería funcionar de la siguiente manera

*Ilustración 8:Funcionamiento general de la API*

Imagen que contiene Diagrama

Descripción generada automáticamente

El funcionamiento sería el siguiente en simples palabras, ignorando las autenticaciones.

### 2.5.1.- Extracción de Información

La información histórica la cual es la fuente estática es pasada a la base de datos. De la misma forma, la información recaudada por el web scraping se irá añadiendo constantemente a la API.

### 2.5.2.- Consulta de datos

El usuario realizará un Login, consultará entre las operaciones que hay, despliegue de estaciones o búsqueda de estaciones en específico. Se realizará la consulta a la base de datos y retornará una respuesta según la operación consultada.

En caso de que el usuario haga Login pero no quiera consultar nada, se termina la operación

## 2.6.- Funcionamiento de la API

Evitando la prolongación de este informe, se creó un archivo README.md en el repositorio de GITHUB que tiene la explicación del cómo se debe ejecutar el programa.

# 3.- Tecnología utilizada

## 3.1.- Lenguaje utilizado

Se utilizó NodeJS, ya que los integrantes del grupo tienen más dominio en aquel lenguaje. Por temas de tiempo no se pudo permitir explorar funcionalidades por otros lenguajes de programación.

Para realizar la API REST, se utilizó NodeJS con PostgreSQL como base de datos.

## 3.2.- Máquina Virtual ORACLE VM Virtualbox

Para el desarrollo del proyecto se instaló la máquina virtual proporcionada por Oracle, VirtualBox, dentro de ella instalamos una distribución de Linux basado en Ubuntu, para ser más específico Lubuntu 20.04 Lts.

## 3.3.- Dependencias de NodeJS

### 3.3.1.- Cheerio

Cheerio es una dependencia que nos permite realizar Web Scraping vía NodeJS. Éste toma con las funciones find( “parámetro” ) el permite buscar dentro del HTML del URL investigado, encontrando la información que posee tal parámetro. También tiene funciones para navegar alrededor del código HTML como con la función next(), parent(), children(). También con la función text(), permite sólo tomar la información en formato texto.

### 3.3.2.- Bcryptjs

Es una herramienta que nos permite realizar encriptaciones (como dice su nombre), dentro del proyecto se pudo utilizar para encriptar las contraseñas del usuario.

### 3.3.3.- CORS

Es un paquete que permite la comunicación entre servidores/dominios, en este caso realiza más sencillo las funciones HTTP con el proyecto.

### 3.3.4.- DotenV

Es un gestor de variables de entorno como configuración, en nuestro caso fue utilizado para manejar bien la temática del puerto del localhost y manejar el tema de la SECRETKEY

### 3.3.5.- Express

Es un marco de aplicación web de back-end para Node.js, en simples palabras, es un framework web escrito en Javascript y alojado dentro del entorno de ejecución Node.js. El uso de éste es bastante importante para este proyecto ya que define éste como la API-REST

### 3.3.6.- JsonWebToken

Es un estándar qué está dentro del documento, en la práctica se trata de una cadena de texto que tiene tres partes codificadas en Base64. Éste funciona como token para poder ser utilizado en algún otro momento. La firma se construye de tal forma que vamos a poder verificar que el remitente es quien dice ser, y que el mensaje no se ha modificado por el camino. El funcionamiento ya fue explicado en el punto 2.3.

### 3.3.7.- Morgan

Es un middleware que nos ayudará a identificar a los clientes que acceden a nuestra aplicación. Básicamente es un registrador.

### 3.3.8.- Node

Node, a pesar de que fue mencionado anteriormente en el punto 3.1, existen muchas variaciones de Node por sus versiones. Y también, antes o después de ciertas versiones, cambian ciertas características o funcionalidades. Por ejemplo después de Node v.13, importar o exportar funciones ya sea de las dependencias o de instancias que fueron creadas para el proyecto es relativamente más sencillo de entender. Hay muchos cambios de sintaxis que ayudaron a que el trabajo sea más fácil de sobrellevar.

### 3.3.9.- PG

Node y PostgreSQL son dos estancias distintas que requieren una conexión, gracias a PG, podemos conectar nuestro proyecto de Node.js a la base de datos de PostgreSQL. Básicamente esa es la funcionalidad de PG

### 3.3.10.- Nodemon

Nodemon es una dependencia de tipo dev, la cual solo es usada y requerida al momento de escribir el código. Su principal función es monitorear los cambios realizados al código fuente que se está desarrollando y reinicia automáticamente el servidor.

## 3.4.- Uso de PostgreSQL

Es un sistema de gestión de base de datos, para poder ocuparlo y trabajar con Postgresql en nuestro código, es que instalamos los módulos *Express y pg*, que nos permiten poder gestionar la base de datos creada. A la cual gracias a los métodos POST, GET, PUT, DELETE es consultada para el ingreso, la obtención, modificación y eliminación de los datos requeridos.

## 3.5.- Uso de Github

Github es un sistema de gestión de versiones en donde los desarrolladores pueden administrar sus proyectos, ordenando el código de cada una de las nuevas versiones. Una de sus principales características son sus funciones colaborativas las que ayudan a que todos los contribuidores aporten sus conocimientos en base a el proyecto a realizar.

## 3.6.- Uso de Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y MacOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. Fue utilizado durante todo el proyecto cómo el editor de código principal por las personas de este equipo.

## 3.7.- Uso de Postman

Postman es una aplicación la cual nos permite realizar pruebas a APIS. Sus principales características son la capacidad de realizar pruebas con protocolos HTTP a través de una interfaz gráfica amigable con el usuario. De esta manera podemos enviar y recibir diferentes tipos de respuestas que pueden ser posteriormente validadas.

# 4.- Conclusión

A pesar de que el proyecto no funcionó como teníamos planeado, en lo más mínimo. Cumplimos en parte con los objetivos planteados para este proyecto, en el proceso de prueba y error que pasó el trabajo, logramos:

* Comprender el funcionamiento del protocolo HTTP (sus verbos y estados): Cuando realizamos diferente tipo de pruebas, el funcionamiento del protocolo HTTP si nos avisó cuáles eran nuestros errores, otro tema fue que no logramos solucionar a tiempo aquellos problemas.
* Comprender el funcionamiento de aplicaciones stateless, mecanismos asíncronos, modelamiento de base de datos, diseño de sistema, codificación eficiente de algoritmos y funcionamiento REST: Quizás el 50% de este objetivo fue logrado, dentro del lenguaje trabajado, usar mecanismos asíncronos era primordial ya que ciertas funciones podrían tomar más tiempo de lo requerido y no por eso el programa tenía que detenerse, el modelamiento de datos no fue el más óptimo, el diseño de la solución y sistema intentó llevarse a cabo de la mejor manera posible y el tema del funcionamiento REST fue aplicado de una manera simple para consumir.