

Universidad Mariano Galvez De Guatemala  
Faculta De Ingenieria  
Seccion A



PROYECTO I  
ip: 45.77.111.103  
contraseña: desarrollo2024

Carlos Andrés Ramírez García	7690-21-10603
Fernando Omar López Morales	7690-21- 2075
Bairon Ismael Castellanos Valle	7690-21-10973

DESARROLLO WEB

## Proyecto I

### Desarrollo Web

El proyecto consiste en realizar un sistema de marcaje web, es decir, tomar la hora de entrada y salida de un empleado. El proyecto debe estar dividido en 2 partes a los que se llamará frontend y backend. Para el frontend no hay restricción de la tecnología utilizar pero debe contener JavaScript, CSS3 y HTML5 y para el backend Spring Boot Java, el sitio debe ejecutarse sobre un Servidor Privado Virtual (VPS), por lo que deberán contratar el servicio al menos para el día de la presentación.

#### # Servidor Privado Virtual (VPS)

---

#### ## ¿Qué es un VPS?

Un **Servidor Privado Virtual (VPS)** es una tecnología de virtualización que permite dividir un servidor físico en varios servidores virtuales independientes. Cada VPS actúa como un servidor dedicado, con su propio sistema operativo, recursos y configuraciones, aunque todos comparten el mismo hardware físico.

---

#### ## Características Principales

- **Aislamiento**: Cada VPS opera de manera independiente, lo que significa que los problemas en un VPS no afectan a los demás.
- **Recursos Dedicados**: Cada VPS tiene asignados recursos específicos (CPU, RAM, almacenamiento) que no se comparten con otros VPS.
- **Control Total**: Los usuarios tienen control total sobre su VPS, incluyendo la instalación de software, configuraciones y administración del sistema operativo.
- **Escalabilidad**: Los recursos de un VPS se pueden ajustar fácilmente según las necesidades, permitiendo la escalabilidad a medida que la demanda crece.
- **Costo Efectivo**: Ofrece una solución intermedia entre el alojamiento compartido y un servidor dedicado, combinando flexibilidad y costo razonable.

---




#### ## ¿Cómo Funciona?

1. **Virtualización**:
  - Un servidor físico (anfitrión) utiliza software de virtualización para crear múltiples entornos virtuales.
2. **Asignación de Recursos**:
  - Cada VPS recibe una porción de los recursos del servidor físico, como CPU, memoria y almacenamiento.
3. **Aislamiento**:
  - Cada VPS opera de manera independiente con su propio sistema operativo, permitiendo una mayor seguridad y personalización.



---






#### ## (Vultr, en Ubuntu)

## 1. \*\*Virtualización\*\*:

Cloud Compute						Location	Search	+ Deploy
<input type="checkbox"/>	Name	OS	Location	Charges	Status			
<input type="checkbox"/>	proyecto 4096.00 MB AMD High Performance - 45.77.111.103		 New Jersey	\$3.79	 Running	***		

## 2. \*\*Asignación de Recursos\*\*:

 proyecto  
45.77.111.103 New Jersey Created 2 days ago  
[Add Tag +](#)





Overview Usage Graphs Settings Snapshots Backups User-Data Tags DDOS


Bandwidth Usage  
0.16GB

vCPU Usage  
3%



Current Charges  
\$3.79

Location:  New Jersey

IP Address: 45.77.111.103 

IPv6 Address: 2001:19f0:1000:677b:5400:05ff:fe13:429a 


Username: root

Password: .....  

vCPU/s: 2 vCPUs

RAM: 4096.00 MB

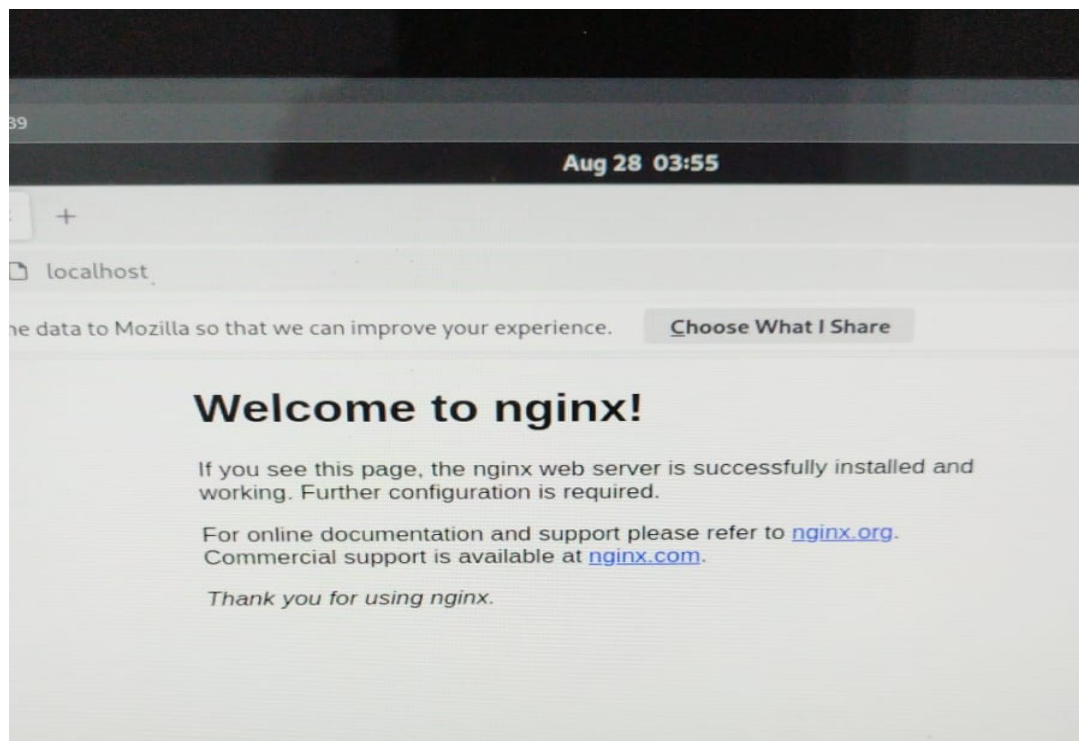
Storage: 100 GB NVMe

Bandwidth: 0.16 GB 

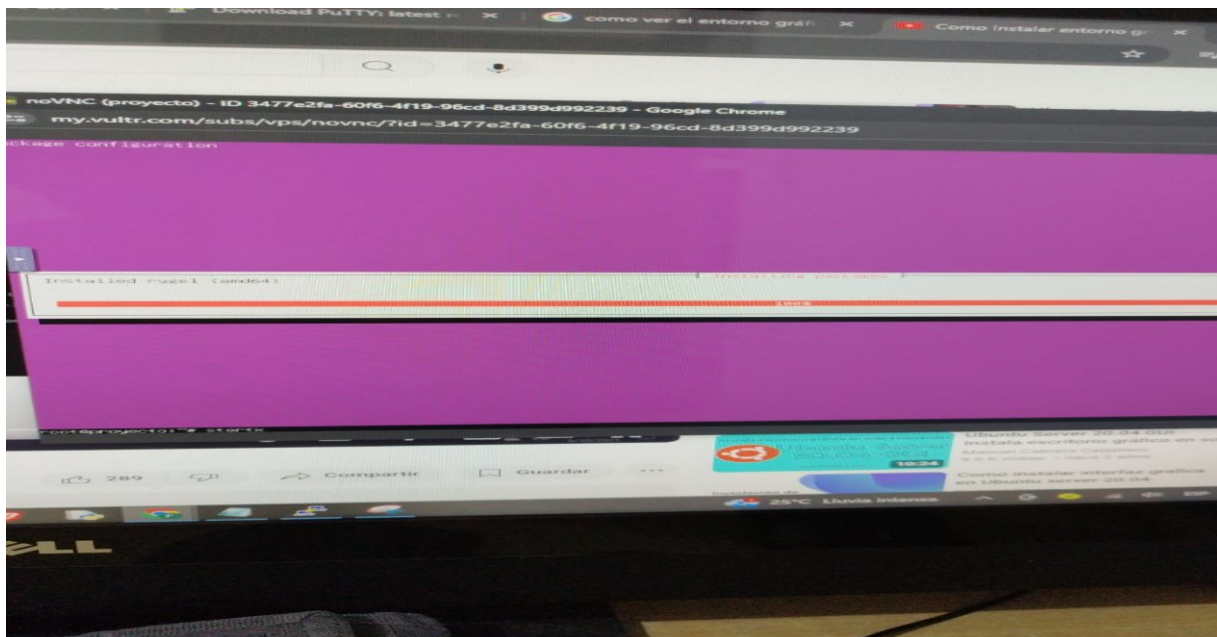
Label: proyecto

OS: Ubuntu 24.04 LTS x64

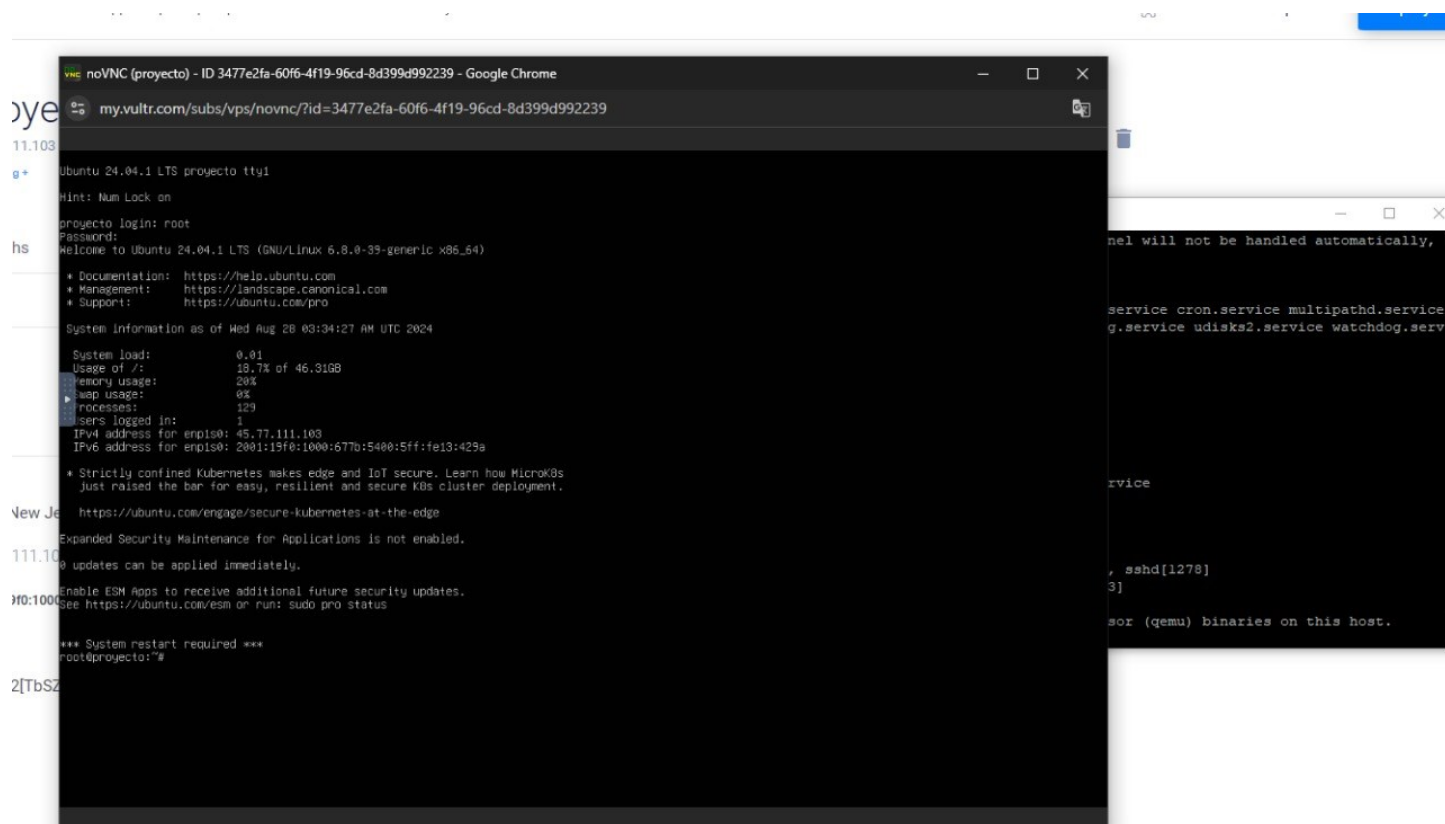
## 3. \*\*incio\*\*:



## # instalacion



## # configuracion



## # Sistema de Marcaje Web

---

### ## ¿Qué es un sistema de marcaje web?

- Un sistema para registrar las **horas de entrada y salida** de los empleados.
- Facilita el seguimiento de asistencia y ayuda a calcular las horas trabajadas.
- Proporciona un acceso fácil y seguro a los registros, tanto para empleados como para administradores.

---

### ## ¿Por qué necesitamos un sistema así?

- **Automatización**: Elimina la necesidad de registros manuales.
- **Precisión**: Reduce errores y asegura que los datos sean exactos.
- **Acceso Remoto**: Permite a los empleados marcar su entrada y salida desde cualquier lugar.

---

### ## ¿Cómo funciona el sistema?

1. **Registro de Entrada**:
  - El empleado ingresa al sistema y registra su hora de entrada con un solo clic.
2. **Registro de Salida**:
  - Al finalizar su jornada, el empleado marca su salida de la misma manera.
3. **Generación de Informes**:
  - El sistema guarda todos los registros y puede generar informes detallados de las horas trabajadas.

---

### ## ¿Qué necesitamos para construirlo?

- **Interfaz de usuario amigable**: Para facilitar el uso diario.
- **Base de datos segura**: Para almacenar los registros de manera fiable.
- **API robusta**: Para manejar todas las solicitudes de registro de tiempo.

---

### ## Tecnologías que vamos a usar

- **HTML, CSS y JavaScript** para el front-end.
- **Node.js y Express** para el back-end.
- **MongoDB** para la base de datos.
- **Git** para control de versiones y colaboración.

---

## ## Beneficios para todos

- **\*\*Para los empleados\*\***: Transparencia en el registro de sus horas de trabajo.
- **\*\*Para los administradores\*\***: Herramientas eficaces para gestionar el tiempo de manera más eficiente.
- **\*\*Para la empresa\*\***: Mejora en la gestión del tiempo y la productividad.

## # Controlador de Autenticación en Spring Boot

---

## ## Introducción

El siguiente código muestra un controlador básico en una aplicación Spring Boot. Este controlador es parte del paquete `com.proyecto1.controller` y está diseñado para manejar las solicitudes relacionadas con la autenticación.

---

## ## Código del Controlador

```
``java
package com.proyecto1.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class AuthController {

}
```

## # Controlador de Índice en Spring Boot

---

### ## Introducción

El siguiente código muestra un controlador en una aplicación Spring Boot diseñado para manejar las solicitudes a la ruta `/index`. Este controlador devuelve la vista correspondiente a esa ruta.

---

### ## Código del Controlador

```
```java
package com.proyecto1.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class IndexController {
    @GetMapping("/index")
    public String index() {
        return "index"; // Esto debe mapear a src/main/resources/templates/index.html
    }
}
```

# Controlador de Login en Spring Boot

---

## Introducción

Este código muestra un controlador en una aplicación Spring Boot diseñado para manejar las solicitudes de inicio de sesión. El controlador gestiona la visualización de la página de inicio de sesión.

---

## Código del Controlador

```
```java
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

@Controller
public class LoginController {

    @GetMapping("/login")
    public String loginPage() {
        return "login"; // Esto debería corresponder a un archivo en src/main/resources/templates/login.html
    }
}
```



## # Controlador de Marcación en Spring Boot

---

### ## Introducción

El siguiente código muestra un controlador en una aplicación Spring Boot que se encarga de gestionar las solicitudes relacionadas con la marcación de tiempo. Este controlador utiliza un servicio para realizar la lógica de negocio.

---

### ## Código del Controlador

```
``java
package com.proyecto1.controller;

import com.proyecto1.service.MarcacionService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class MarcacionController {

    @Autowired
    private MarcacionService marcacionService;

    // Métodos del controlador
}
```

---

# Modelo de Empleado en Spring Boot

---

## ## Introducción

El siguiente código define una entidad `Empleado` en una aplicación Spring Boot, utilizando JPA (Java Persistence API) para gestionar la persistencia de datos en una base de datos.

---

## ## Código del Modelo

```
```java
package com.proyecto1.model;

import jakarta.persistence.*;

@Entity
@Table(name = "empleados")
public class Empleado {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nombre;
    private String username;
    private String contrasenia;

    public void setId(Long id) {
        this.id = id;
    }

    public Long getId() {
        return id;
    }
    // Otros campos y métodos...
}
```

## # Modelo de Marcación en Spring Boot

---

### ## Introducción

El siguiente código define una entidad `Marcacion` en una aplicación Spring Boot, utilizando JPA (Java Persistence API) para gestionar la persistencia de datos en la base de datos. Esta entidad representa una entrada o salida registrada por un empleado.

---

### ## Código del Modelo

```
```java
package com.proyecto1.model;

import jakarta.persistence.*;

@Entity
@Table(name = "marcaciones")
public class Marcacion {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "empleado_id", nullable = false)
    private Empleado empleado;

    private java.sql.Date fecha;
    private java.sql.Time hora;
    private String tipo;

    // Getters y setters
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public Empleado getEmpleado() {
        return empleado;
    }

    public void setEmpleado(Empleado empleado) {
        this.empleado = empleado;
    }
}
```

```

public java.sql.Date getFecha() {
    return fecha;
}

public void setFecha(java.sql.Date fecha) {
    this.fecha = fecha;
}

public java.sql.Time getHora() {
    return hora;
}

public void setHora(java.sql.Time hora) {
    this.hora = hora;
}

public String getTipo() {
    return tipo;
}

public void setTipo(String tipo) {
    this.tipo = tipo;
}
}

```

### ## Conclusión

- Un sistema de marcaje web puede transformar la forma en que gestionamos el tiempo de trabajo.
- Es una solución **eficiente**, **precisa** y **fácil de usar** que beneficia a todos los involucrados.

---