

Práctica 2

Taller de proyecto II

Meroni, Milton Alberto; Marón, Gastón Ariel

Contacto: miltonmeroni@gmail.com, gastonmaron@gmail.com

Facultad de Informática, UNLP

Fecha de entrega: 19 de Octubre de 2015

Contenido

Introducción	1
Instalación de los módulos principales	1
Node.js	2
Npm.....	3
Instalación de los paquetes para el Node.js	4
Express.....	4
Body-parser.....	5
EmailJS	5
Mysql	5
EJS	6
Instalación de servidor LAMP	6
Instalar Apache	7
Instalar MySQL	8
Instalar PHP	11
Instalación de módulos PHP	12
Instalación de PhpMyAdmin	12
Diseño de la base de datos	17
Importación de la base de datos al phpmyadmin	18
Sistema WEB.....	21
Páginas WEB	22
Envío de emails.....	31
Uso del microcontrolador y uso de sensores y actuadores	32
Programa en lenguaje C	32
Programa principal en Node.js.....	33
Posibles errores	34
Comunicación con la PC.....	35
Posibles errores	40
Diferencias entre sistema real y simulado	41
Concurrencia	41

Introducción

Para la simulación del proyecto propuesto, se buscó la mejor forma de poder trasladar las características del Sistema Operativo que tiene la placa Intel Galileo (Linux Yocto) a un Sistema Operativo que pueda ser instalado en una PC. Para ello, se instalaron los módulos que el Yocto necesita en una máquina virtual con un Sistema Operativo Ubuntu.

Los módulos que difieren entre ambos sistemas operativos son:

- Node.js
- Npm

Se debe aclarar que en el proyecto, se mencionarán las instalaciones de otros módulos pero que deberán ser instalados también en la placa. Entre ellos se encuentran:

- Express
- Body-parser
- EmailJS
- Mysql
- EJS

Además, para poder llevar a cabo la simulación, se explicará cómo se debe instalar un servidor LAMP (Linux, Apache, Mysql, Php), utilizado para poder manipular la base de datos.

Instalación de los módulos principales

Antes que nada, para tener una instalación limpia y sin problemas con los privilegios de administrador o super usuario, es recomendable que se cree una carpeta en el directorio del usuario actual. Sin embargo, es necesario tener una contraseña para el super usuario o para poder realizar acciones que impliquen privilegios administrativos, por ejemplo, para ejecutar el comando *sudo*. En este caso el nombre del usuario es “ubuntu”.

```
ubuntu@ubuntu:~$ mkdir /home/ubuntu/practica2
```

El siguiente paso, aunque no es necesario, es dirigirse a la carpeta y desde allí comenzar la instalación:

- `cd /home/Ubuntu/practica2`

Luego, se deben actualizar los repositorios propios de Ubuntu con el comando:

- `sudo apt-get update`

```
ubuntu@ubuntu:~$ cd /home/ubuntu/practica2/  
ubuntu@ubuntu:~/practica2$ sudo apt-get update
```

Una vez que el comando devuelve el prompt, se estará en condiciones para instalar los módulos Node.js y Npm.

Node.js

Para realizar la instalación del programa Node.js se debe ejecutar el siguiente comando:

- `sudo apt-get install Node.js`

Cuando haga click en “Enter”, se iniciará el programa de instalación. Cuando pregunte por si quiere continuar, deberá poner la letra “Y” (en referencia a *yes*) y nuevamente, apretar la tecla “Enter”.

```
ubuntu@ubuntu:~/practica2$ sudo apt-get install nodejs  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following extra packages will be installed:  
  libc-ares2 libv8-3.14.5  
The following NEW packages will be installed:  
  libc-ares2 libv8-3.14.5 nodejs  
0 upgraded, 3 newly installed, 0 to remove and 271 not upgraded.  
Need to get 1,912 kB of archives.  
After this operation, 7,538 kB of additional disk space will be used.  
Do you want to continue? [Y/n] Y
```

Para verificar la correcta instalación del Node.js, se averiguará cual es la versión con el comando:

- `Node.js -v`

```
ubuntu@ubuntu:~/practica2$ nodejs -v  
v0.10.25
```

En este caso, la versión instalada es la v0.10.25.

Npm

Ahora se debe instalar el manejador de paquetes para Node.js, npm. El comando es:

- `sudo apt-get install npm`

```
ubuntu@ubuntu:~/practica2$ sudo apt-get install npm
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  gyp javascript-common libc-ares-dev libjs-jquery libjs-node-uuid
  libjs-underscore libssl-dev libssl-doc libssl1.0.0 libv8-3.14-dev node-abbrev
  node-ansi node-ansi-color-table node-archy node-async node-block-stream
  node-combined-stream node-cookie-jar node-delayed-stream node-forever-agent
  node-form-data node-fstream node-fstream-ignore node-github-url-from-git
  node-glob node-graceful-fs node-gyp node-inherits node-ini
  node-json-stringify-safe node-lockfile node-lru-cache node-mime
  node-minimatch node-mkdirp node-mute-stream node-node-uuid node-nopt
  node-normalize-package-data node-npmlog node-once node-osenv node-qs
  node-read node-read-package-json node-request node-retry node-rimraf
  node-semver node-sha node-sigmund node-slide node-tar node-tunnel-agent
  node-underscore node-which nodejs-dev zlib1g-dev
Suggested packages:
  apache2 lighttpd httpd node-hawk node-aws-sign node-oauth-sign
  node-http-signature
The following NEW packages will be installed:
  gyp javascript-common libc-ares-dev libjs-jquery libjs-node-uuid
  libjs-underscore libssl-dev libssl-doc libv8-3.14-dev node-abbrev node-ansi
  node-ansi-color-table node-archy node-async node-block-stream
  node-combined-stream node-cookie-jar node-delayed-stream node-forever-agent
  node-form-data node-fstream node-fstream-ignore node-github-url-from-git
  node-glob node-graceful-fs node-gyp node-inherits node-ini
  node-json-stringify-safe node-lockfile node-lru-cache node-mime
  node-minimatch node-mkdirp node-mute-stream node-node-uuid node-nopt
  node-normalize-package-data node-npmlog node-once node-osenv node-qs
  node-read node-read-package-json node-request node-retry node-rimraf
  node-semver node-sha node-sigmund node-slide node-tar node-tunnel-agent
  node-underscore node-which nodejs-dev npm zlib1g-dev
The following packages will be upgraded:
  libssl1.0.0
1 upgraded, 58 newly installed, 0 to remove and 270 not upgraded.
Need to get 4,438 kB of archives.
After this operation, 16.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Nuevamente, poner la letra “Y” y apretar “Enter” para su instalación. Para corroborar que está instalado en npm, se averiguará cuál es su versión:

- `npm -v`

```
ubuntu@ubuntu:~/practica2$ npm -v
1.4.21
```

Con esta instalación terminada, ambos sistemas operativos están en las mismas condiciones.

Instalación de los paquetes para el Node.js

La instalación de estos paquetes se realizará a través del manejador de paquetes npm. Para la correcta instalación de estos paquetes, NECESARIAMENTE se debe estar en la carpeta donde se va a trabajar con el Node.js. La carpeta propuesta en este informe es la que se encuentra en la siguiente ruta:

- */home/ubuntu/practica2*

Una vez dentro del directorio se procederá a la instalación de los paquetes necesarios para la simulación.

Express

“Express” es un framework web rápido y minimalista para Node.js que proporciona un conjunto robusto de características para aplicaciones web, con una gran variedad de métodos de utilidad HTTP.

Para la instalación se debe ejecutar el siguiente comando:

- *npm install express*

```
express@4.13.3 node_modules/express
├── escape-html@1.0.2
├── merge-descriptors@1.0.0
├── array-flatten@1.1.1
├── cookie@0.1.3
├── utils-merge@1.0.0
├── cookie-signature@1.0.6
├── methods@1.1.1
├── fresh@0.3.0
├── range-parser@1.0.2
├── vary@1.0.1
├── path-to-regexp@0.1.7
├── content-type@1.0.1
├── etag@1.7.0
├── parseurl@1.3.0
├── content-disposition@0.5.0
├── serve-static@1.10.0
├── depd@1.0.1
├── qs@4.0.0
├── finalhandler@0.4.0 (unpipe@1.0.0)
├── on-finished@2.3.0 (ee-first@1.1.1)
├── debug@2.2.0 (ms@0.7.1)
├── proxy-addr@1.0.8 (forwarded@0.1.0, ipaddr.js@1.0.1)
├── send@0.13.0 (destroy@1.0.3, ms@0.7.1, statuses@1.2.1, mime@1.3.4, http-errors@1.3.1)
├── accepts@1.2.13 (negotiator@0.5.3, mime-types@2.1.7)
└── type-is@1.6.9 (media-typer@0.3.0, mime-types@2.1.7)
```


Que generará todo el árbol de módulos que contiene el “express”. Cabe destacar que en la carpeta donde se ejecutó el comando se generó una nueva carpeta con el nombre “node_modules” donde serán instalados este y todos los módulos siguientes.

Body-parser

Es el encargado de tomar todos los parámetros en formato JSON, provenientes de un método “post” desde un formulario html.

Para la instalación se debe ejecutar el siguiente comando:

- *npm install body-parser*

```
ubuntu@ubuntu:~/practica2$ npm install body-parser
body-parser@1.14.1 node_modules/body-parser
├── bytes@2.1.0
├── content-type@1.0.1
├── depd@1.1.0
├── qs@5.1.0
├── iconv-lite@0.4.12
├── http-errors@1.3.1 (inherits@2.0.1, statuses@1.2.1)
├── on-finished@2.3.0 (ee-first@1.1.1)
├── raw-body@2.1.4 (unpipe@1.0.0)
├── debug@2.2.0 (ms@0.7.1)
└── type-is@1.6.9 (media-typer@0.3.0, mime-types@2.1.7)
```

EmailJS

Es una herramienta para poder mandar emails, con texto plano o con formato html y archivos adjuntos (archivos) desde un servidor Node.js a cualquier servidor smtp.

Para la instalación se debe ejecutar el siguiente comando:

- *npm install emailjs*

```
ubuntu@ubuntu:~/practica2$ npm install emailjs
emailjs@1.0.0 node_modules/emailjs
├── bufferjs@1.1.0
├── addressparser@0.3.2
├── starttls@0.2.1
├── moment@1.7.0
└── mime-lib@0.2.14 (addressparser@0.2.1, encoding@0.1.11)
```

Mysql

Es un driver de Node.js para utilizar mysql. Con esta herramienta se puede crear una conexión a una base de datos y poder realizar consultas (query).

Para la instalación se debe ejecutar el siguiente comando:

- *npm install mysql*

```
ubuntu@ubuntu:~/practica2$ npm install mysql
mysql@2.9.0 node_modules/mysql
├── bignumber.js@2.0.7
├── readable-stream@1.1.13 (isarray@0.0.1, string_decoder@0.10.31, core-util-is@1.0.1, inherits@2.0.1)
```

EJS

Es una herramienta que permite desarrollar la visual de un código html. Combina datos y un template html para generar páginas web de manera dinámica.

Para la instalación se debe ejecutar el siguiente comando:

- *npm install ejs*

```
ubuntu@ubuntu:~/practica2$ npm install ejs
ejs@2.3.4 node_modules/ejs
```

Si ahora observamos el directorio actual se podrá ver todos los módulos instalados. Si se ejecutan los siguientes comandos:

- *cd node_modules*
- *ls*

Se observará las siguientes carpetas que corresponden a cada uno de los módulos instalados:

```
ubuntu@ubuntu:~/practica2$ cd node_modules/
ubuntu@ubuntu:~/practica2/node_modules$ ls
body-parser ejs emailjs express mysql
ubuntu@ubuntu:~/practica2/node_modules$
```

Instalación de servidor LAMP

Se denomina "LAMP" a un grupo de software de código libre que se instala normalmente en conjunto para habilitar un servidor para alojar sitios y aplicaciones web dinámicas. Este término en realidad es un acrónimo que representa un sistema operativo **L**inux con un servidor **A**pache, el sitio de datos es almacenado en base de datos **M**ySQL y el contenido dinámico es procesado con **P**HP.

Instalar Apache

Podemos instalar Apache fácilmente desde el gestor de paquetes de Ubuntu, apt. El gestor de paquetes nos permite instalar con mayor facilidad un software desde un repositorio conservado por Ubuntu.

- `sudo apt -get update`

```
ubuntu@ubuntu:~$ sudo apt-get update
[sudo] password for ubuntu:
```

Actualiza los repositorios del Sistema Operativo.

- `sudo apt -get install apache2`

```
ubuntu@ubuntu:~$ sudo apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.1-0
Suggested packages:
  apache2-doc apache2-suexec-pristine apache2-suexec-custom
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap liblua5.1-0
0 upgraded, 9 newly installed, 0 to remove and 270 not upgraded.
Need to get 1,541 kB of archives.
After this operation, 6,356 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

Como se mencionó antes, para continuar la instalación, se debe poner la tecla “Y” y luego presionar “Enter”.

Una vez finalizada la instalación del apache, podremos ver si el mismo se instaló correctamente. Para realizar esto, tendremos que conocer el “ip” de nuestro SO. Podemos obtenerlo a través del comando “ifconfig”.

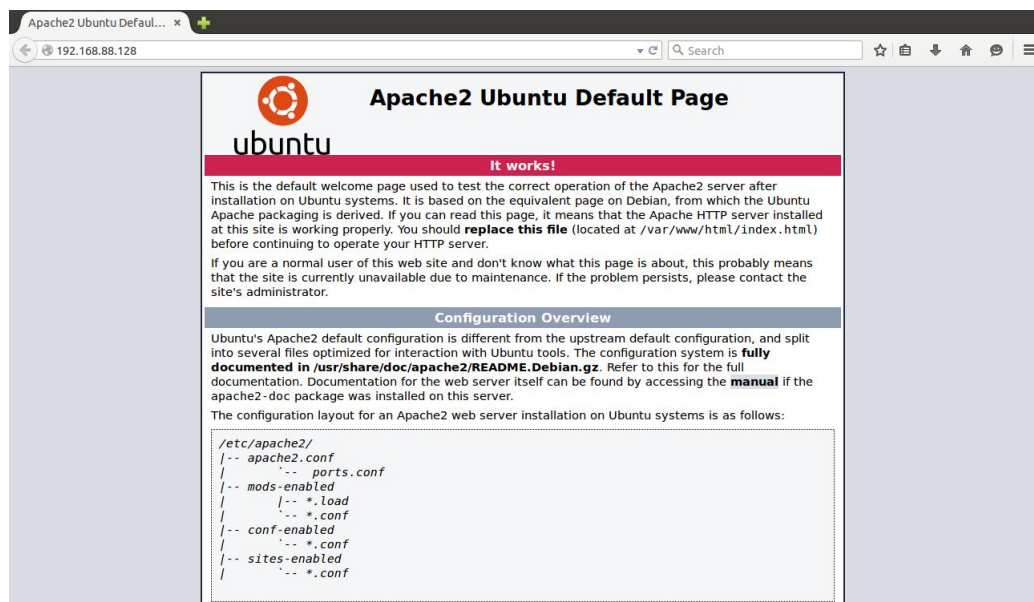
```

ubuntu@ubuntu:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:8e:8b:cd
          inet addr:192.168.88.128  Bcast:192.168.88.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe8e:8bcd/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:69070 errors:0 dropped:0 overruns:0 frame:0
          TX packets:31649 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:90430391 (90.4 MB)  TX bytes:2793238 (2.7 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:2147 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2147 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1034560 (1.0 MB)  TX bytes:1034560 (1.0 MB)

```

Una vez realizado esto, se debe abrir el navegador y poner el “ip” obtenido. Si el servidor apache está corriendo, se verá la siguiente página, lo que indica el correcto funcionamiento.



Instalar MySQL

Ahora que ya tenemos nuestro servidor web configurado y corriendo, es el momento de instalar MySQL.

- `sudo apt-get install mysql-server libapache2-mod-auth-mysql php5-mysql`

MySQL es un sistema de gestión de base de datos. Básicamente, se encarga de organizar y facilitar el acceso a las bases de datos donde nuestro sitio puede almacenar información.

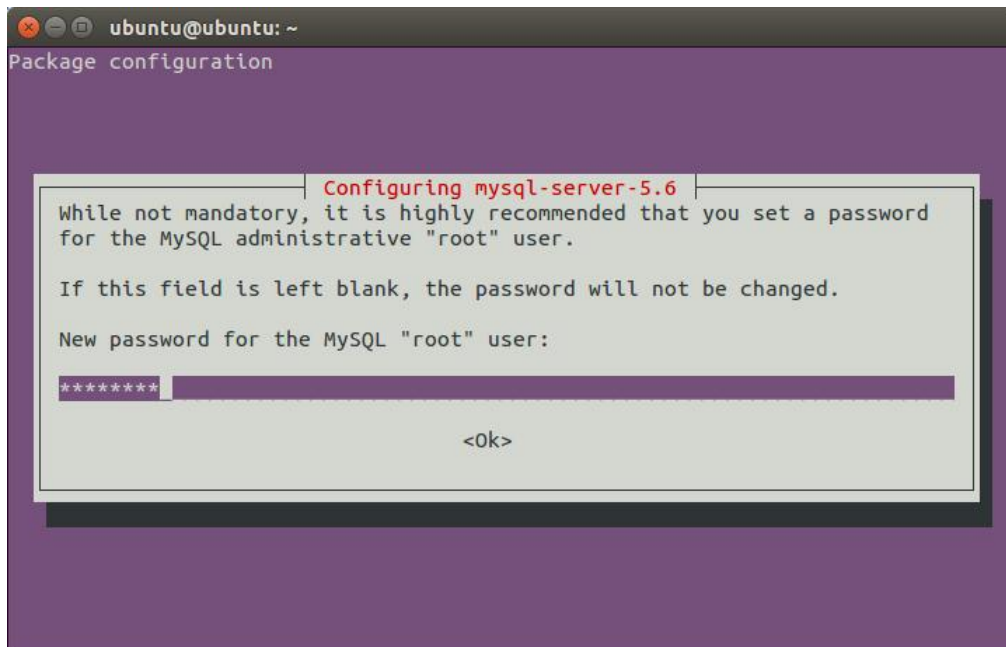
```

ubuntu@ubuntu:~$ sudo apt-get install mysql-server libapache2-mod-auth-mysql php
5-mysql
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libaio1 libdbd-mysql-perl libdbi-perl libhtml-template-perl libmysqlclient18
  libterm-readkey-perl mysql-client-5.6 mysql-client-core-5.6 mysql-common
  mysql-server-5.6 mysql-server-core-5.6 php5-common
Suggested packages:
  libmldbm-perl libnet-daemon-perl libsql-statement-perl
  libipc-sharedcache-perl mailx tinyca php5-user-cache
The following NEW packages will be installed:
  libaio1 libapache2-mod-auth-mysql libdbd-mysql-perl libdbi-perl
  libhtml-template-perl libmysqlclient18 libterm-readkey-perl mysql-client-5.6
  mysql-client-core-5.6 mysql-common mysql-server mysql-server-5.6
  mysql-server-core-5.6 php5-common php5-mysql
0 upgraded, 15 newly installed, 0 to remove and 270 not upgraded.
Need to get 21.2 MB of archives.
After this operation, 156 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y

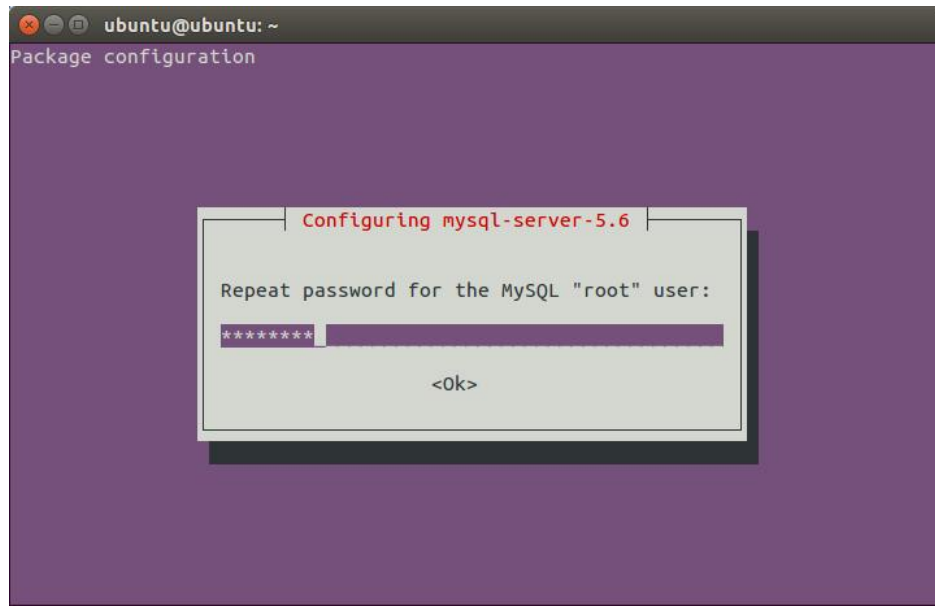
```

Tal como ya realizamos antes, debemos presionar la tecla “Y” y luego presionar “Enter” para continuar.

La instalación, nos pedirá que ingresemos una contraseña para un usuario “root”.



Luego, se debe repetir la clave antes ingresada.



Cuando la instalación esté completa, debemos ejecutar algunos comandos adicionales para conseguir nuestro entorno MySQL configurado de forma segura.

- `sudo mysql_install_db`

```
ubuntu@ubuntu:~$ sudo mysql_install_db
```

El comando antes ingresado, le indica a MySQL que tiene que crear su propia base de datos para la estructura del directorio donde se almacenará la información.

- `sudo /usr/bin/mysql_secure_installation`

```
ubuntu@ubuntu:~$ sudo /usr/bin/mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MySQL
SERVERS IN PRODUCTION USE! PLEASE READ EACH STEP CAREFULLY!

In order to log into MySQL to secure it, we'll need the current
password for the root user. If you've just installed MySQL, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MySQL
root user without the proper authorisation.

You already have a root password set, so you can safely answer 'n'.

Change the root password? [Y/n] n
... skipping.
```


Si no queremos cambiar la contraseña “root”, debemos poner “n” como se muestra en la imagen anterior y siguiendo, presionar la tecla “Enter”.

Luego, en las demás preguntas que nos realice la instalación, se deberá presionar “Y” y luego “Enter”.

```
By default, a MySQL installation has an anonymous user, allowing anyone
to log into MySQL without having to have a user account created for
them. This is intended only for testing, and to make the installation
go a bit smoother. You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] Y
... Success!

Normally, root should only be allowed to connect from 'localhost'. This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] Y
... Success!

By default, MySQL comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] Y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] Y
... Success!

All done! If you've completed all of the above steps, your MySQL
installation should now be secure.

Thanks for using MySQL!
```

Instalar PHP

PHP es el componente de nuestra configuración que procesará código para mostrar contenido dinámico. Puede ejecutar secuencias de comandos, conectarse a nuestras bases de datos MySQL para obtener información, y entregar el contenido procesado a nuestro servidor web para mostrarlo.

- *sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt*

```

ubuntu@ubuntu:~$ sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libmcrypt4 php5-cli php5-json php5-readline
Suggested packages:
  php-pear libmcrypt-dev mcrypt
The following NEW packages will be installed:
  libapache2-mod-php5 libmcrypt4 php5 php5-cli php5-json php5-mcrypt
  php5-readline
0 upgraded, 7 newly installed, 0 to remove and 270 not upgraded.
Need to get 4,636 kB of archives.
After this operation, 20.4 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y

```

Instalación de módulos PHP

Para mejorar la funcionalidad de PHP, podemos instalar opcionalmente algunos módulos adicionales.

En nuestro caso, se instalará el módulo “php5-mysql”.

- *sudo apt-get install php5-mysql*

```

ubuntu@ubuntu:~$ sudo apt-get install php5-mysql
Reading package lists... Done
Building dependency tree
Reading state information... Done
php5-mysql is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 270 not upgraded.

```

Instalación de PhpMyAdmin

PhpMyAdmin nos da la posibilidad de administrar la BD mysql a través de un entorno amigable al usuario.

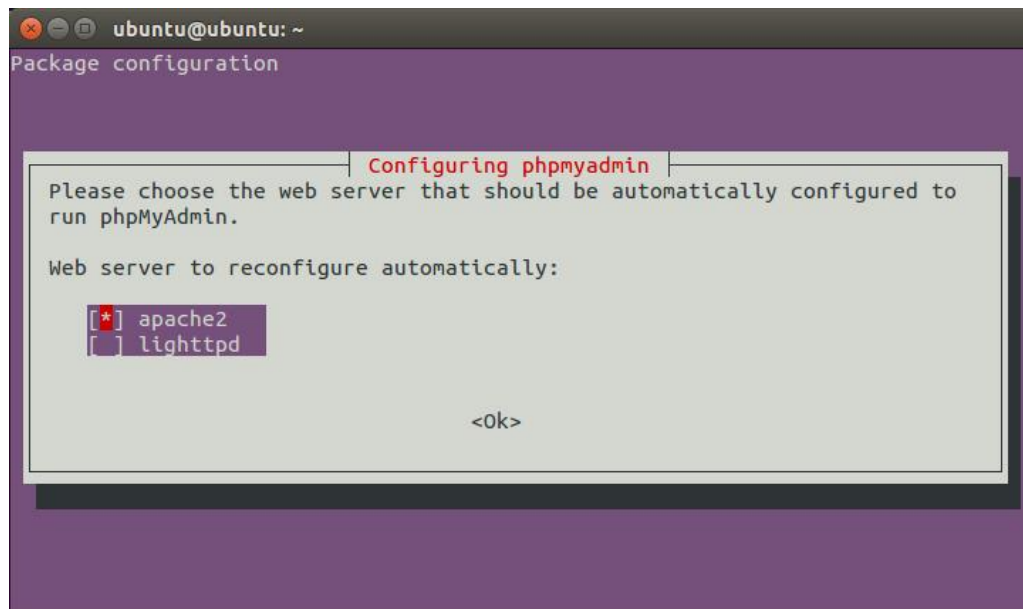
- *sudo apt-get install phpmyadmin*

```

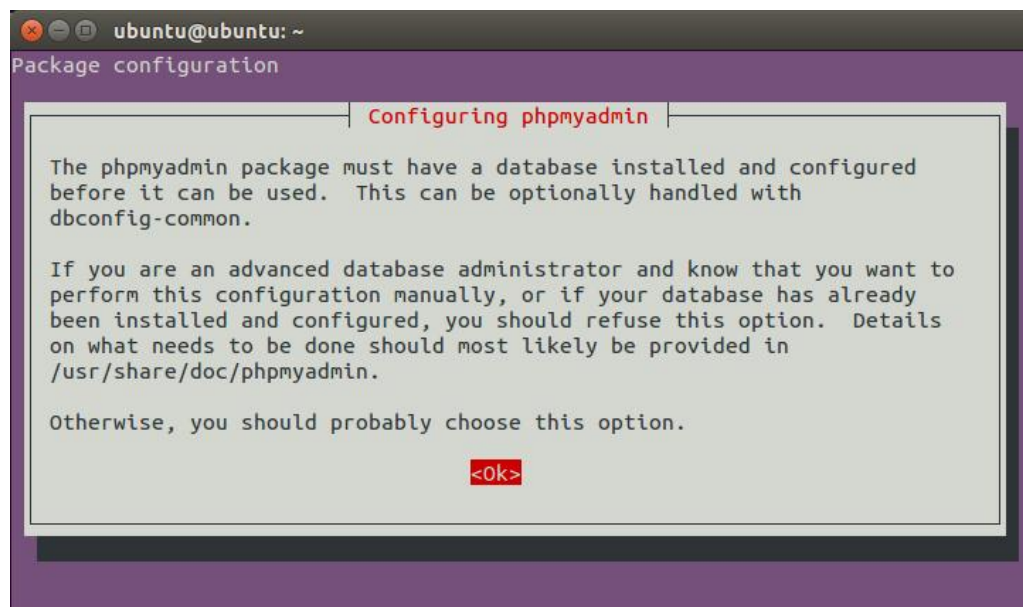
ubuntu@ubuntu:~$ sudo apt-get install phpmyadmin
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  dbconfig-common libjs-sphinxdoc php-gettext php-tcpdf php5-gd
Suggested packages:
  php5-imagick
The following NEW packages will be installed:
  dbconfig-common libjs-sphinxdoc php-gettext php-tcpdf php5-gd phpmyadmin
0 upgraded, 6 newly installed, 0 to remove and 270 not upgraded.
Need to get 12.4 MB of archives.
After this operation, 46.4 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y

```

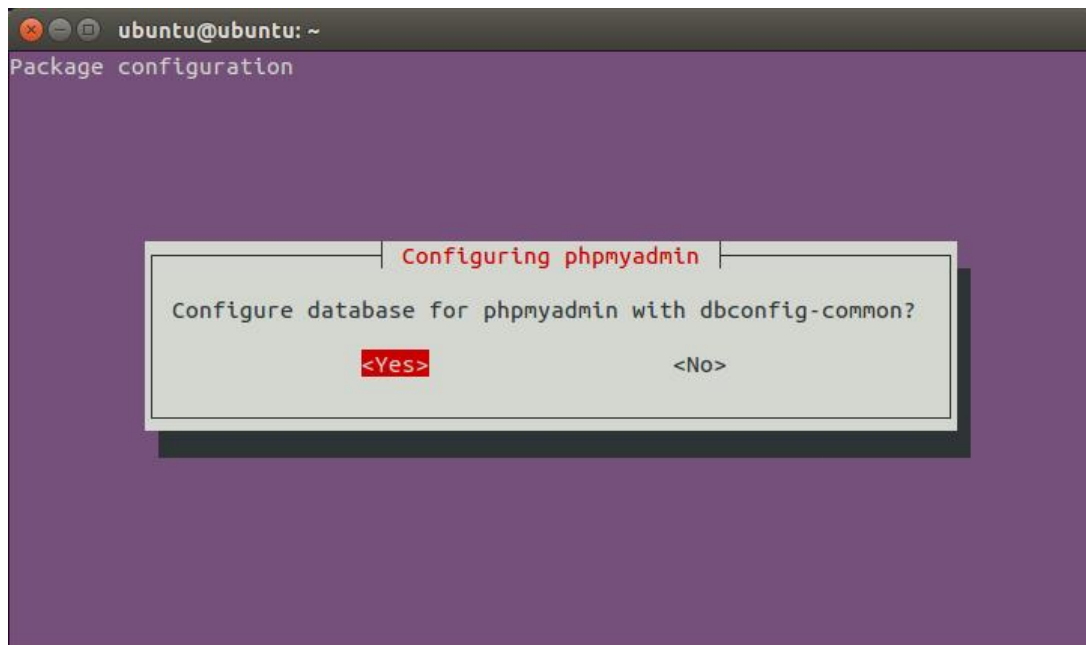

Debemos seleccionar que servidor se debe configurar, en este caso, se seleccionará “apache2” con la barra espaciadora y se presionará “Enter”.



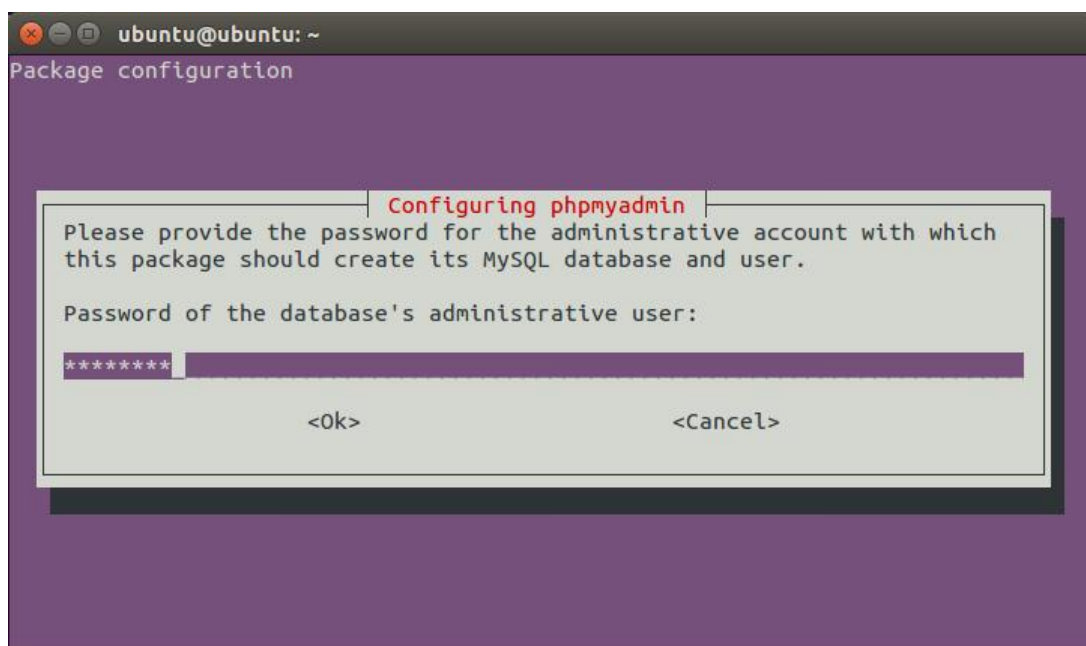
En la imagen siguiente, se realiza la configuración de PhpMyAdmin por lo que se debe presionar “Enter”.



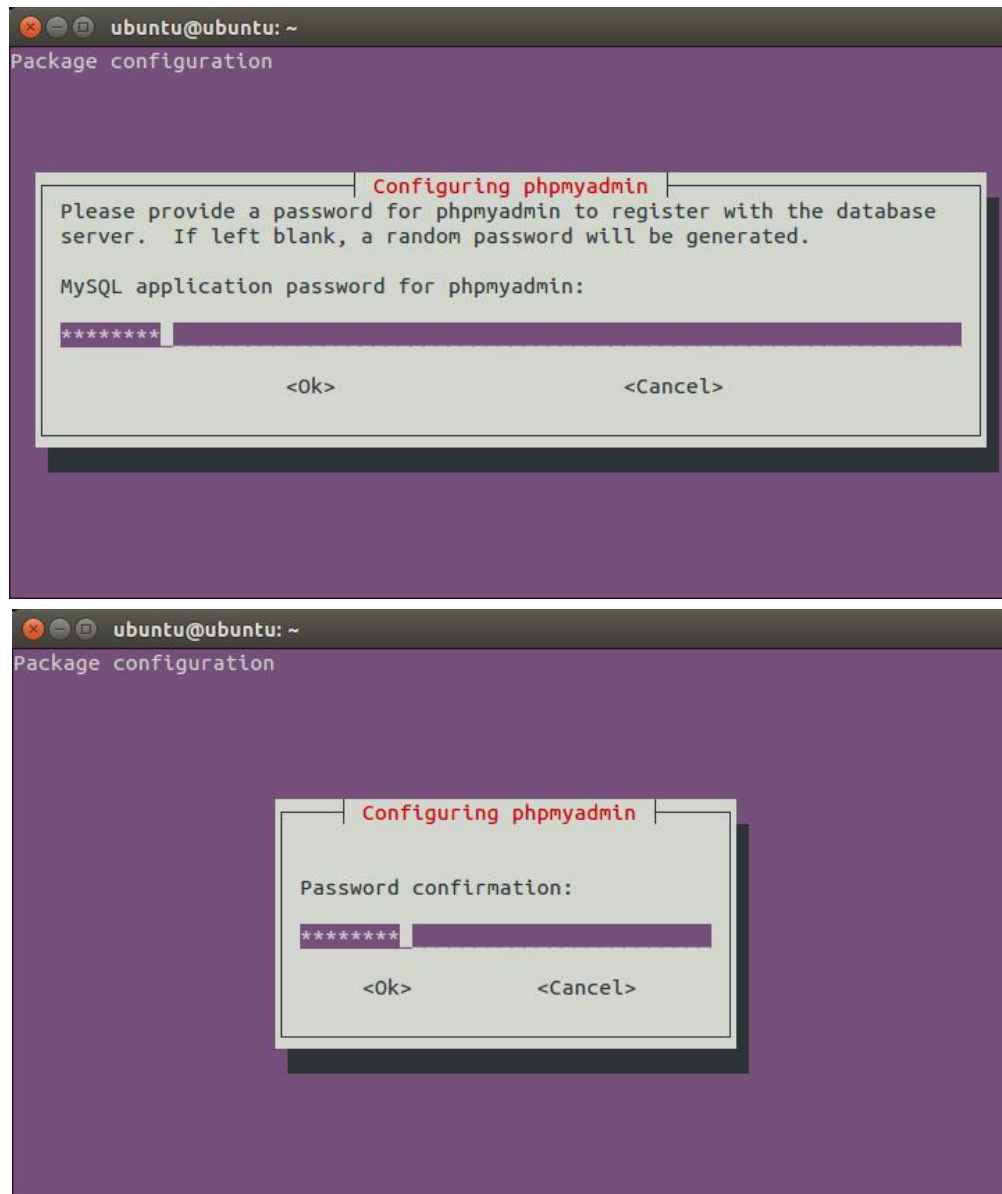
A continuación, se selecciona “Yes” y se presiona “Enter”.



Se le pedirá la contraseña de su administrador de la base de datos.



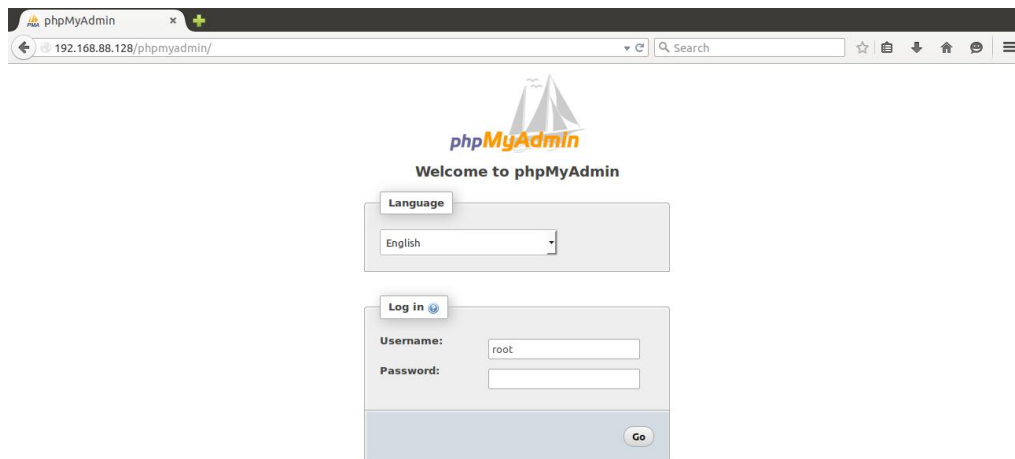
Y finalmente, se le pedirá que elija y confirme una contraseña para la aplicación phpMyAdmin.



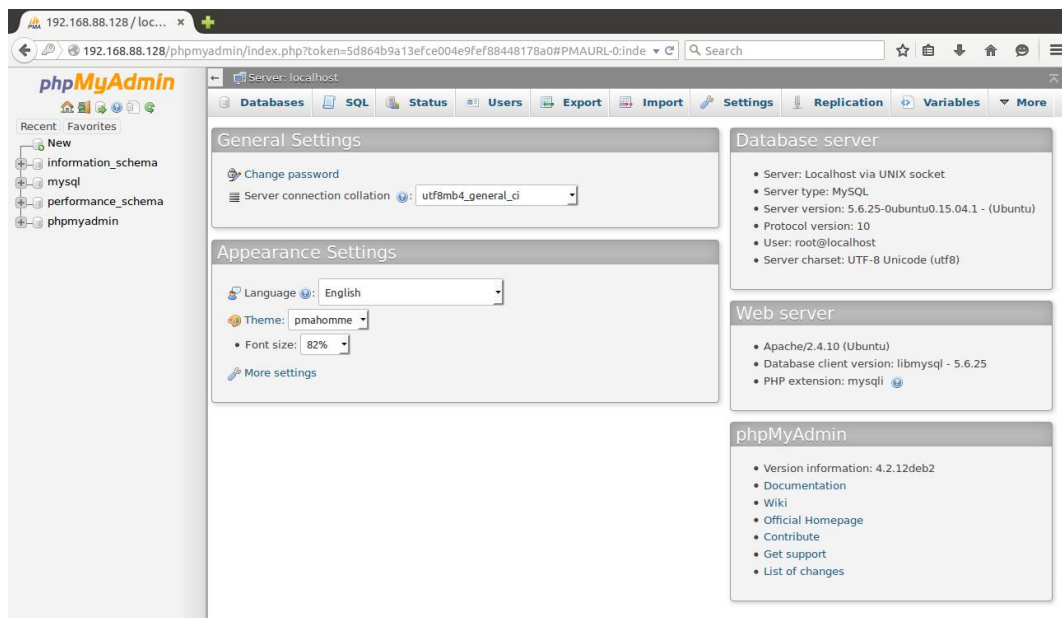
Finalmente, para corroborar el correcto funcionamiento del administrador de base de datos, se debe ingresar en la barra de direcciones del navegador la siguiente dirección:

- *ipDelServidor/phpmyadmin*

Deberá aparecer la siguiente ventana



Para ingresar al sistema, se debe poner el usuario root, junto con la contraseña del servidor de mysql que se proporcionó en la instalación. Una vez que se ingresó, se debe ver la siguiente ventana:



Diseño de la base de datos

La comunicación entre la placa Intel Galileo y la PC será a través de un servidor que aloja la base de datos a utilizar. Para ello se propuso una base de datos que contiene dos tablas Usuario y Auditoria.

La tabla Usuario es aquella que albergará todos los datos del usuario, es por ello que tiene los siguientes atributos:

Atributos de la tabla Usuario	Función	Tipo de dato
id	Id único de la entidad usuario	Int-autoincremental
nombre	Nombre del usuario	Varchar(255)
apellido	Apellido del usuario	Varchar(255)
dni	DNI único del usuario	Varchar(255)
email	Email único del usuario	Varchar(255)
password	Contraseña única del usuario	Varchar(255)
temp	Temperatura elegida por el usuario	Varchar(255)
luz	Porcentaje de luz elegido por el usuario	Varchar(255)

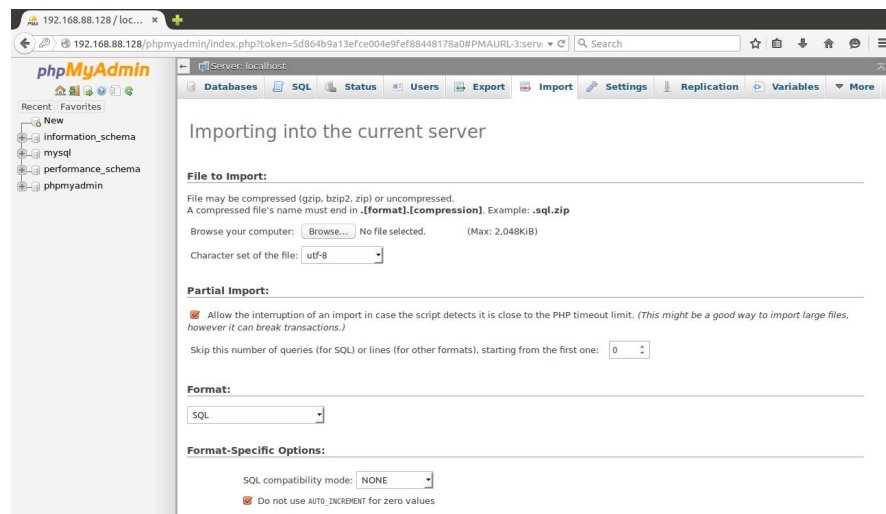
Mientras que la tabla Auditoria es aquella que albergará todos los datos de entrada y salida de los usuarios. Es usada para llevar el control de usuarios activos y un registro de los usuarios que han ingresado a la habitación. Tiene los siguientes atributos:

Atributos de la tabla Auditoria	Función	Tipo de dato
id	id único de cada auditoria	Int-Autoincremental
email	Email del usuario que ingreso a la habitación	Varchar(255)
fechaEntrada	Hora y fecha en la que el usuario ingresó a la habitación.	Varchar(255)
fechaSalida	Hora y fecha en la que el usuario sale de la habitación. Inicia como NULL e indica que el usuario todavía no salió de la habitación.	Varchar(255)

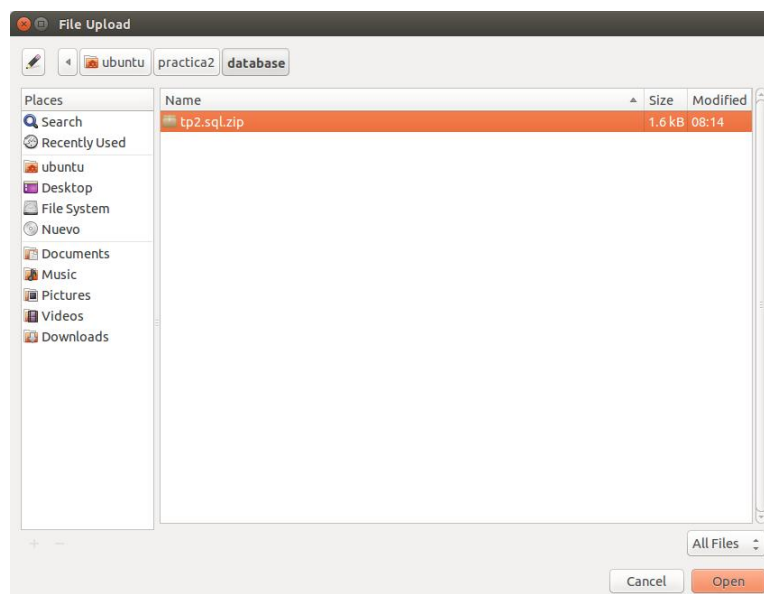
Importación de la base de datos al phpmyadmin

En el caso de que se quiera importar la base de datos sin tener que definir las entidades anteriores, en la carpeta “database”, ubicada en el mismo directorio que este documento, se encuentra un archivo llamado “tp2.sql.zip” que contiene la base de datos “tp2” con las tablas anteriormente mencionadas.

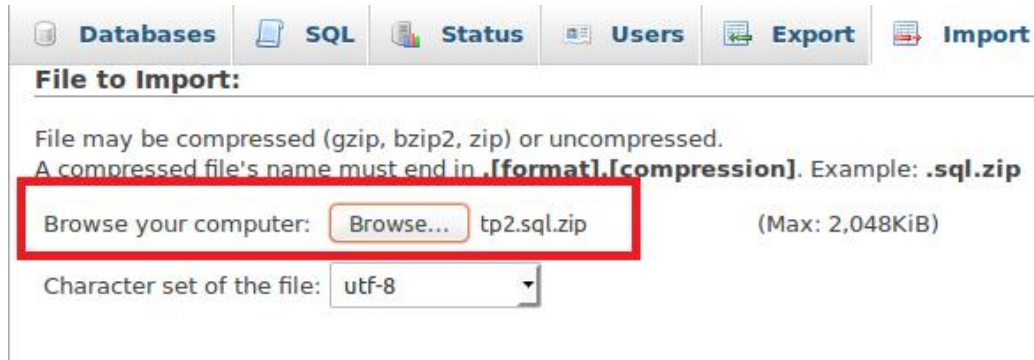
Para importarla, se debe ingresar al sistema “phpmyadmin” y hacer click en la solapa “import”.



Luego, al lado de donde dice “Browse your computer:”, hay un botón, “Browse...” que permite buscar en el sistema de archivos una base de datos. Navegar hasta el directorio que se mencionó y elegir el archivo comprimido.



Una vez escogido, se pondrá el nombre del archivo junto al botón.



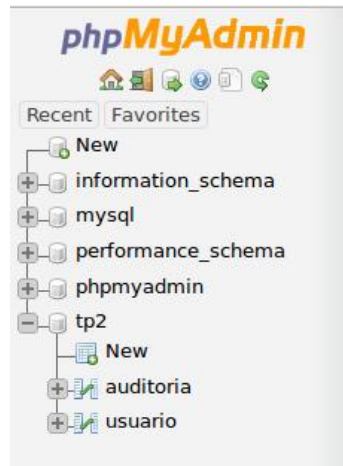
File to Import:

File may be compressed (gzip, bzip2, zip) or uncompressed.
A compressed file's name must end in **.[format].[compression]**. Example: **.sql.zip**

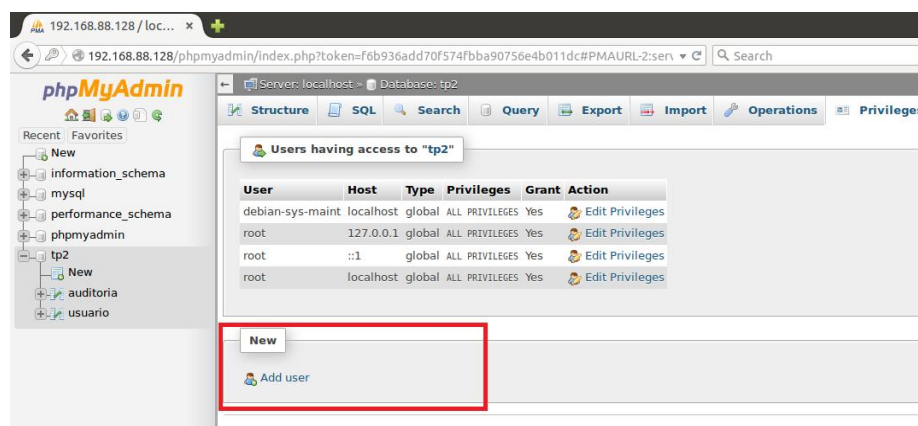
Browse your computer: tp2.sql.zip (Max: 2,048KiB)

Character set of the file:

Finalmente, hacer click en el botón “Go” que está al final de la página de importación. Automáticamente se actualizará la página con la base de datos ya importada y lista para utilizar.

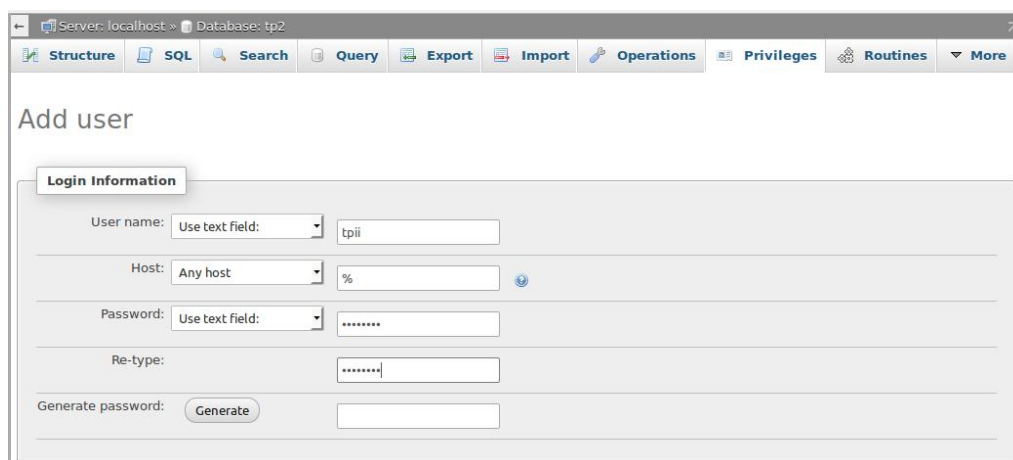


Para poder acceder a esta nueva base de datos desde la aplicación, se debe crear un usuario con ciertos privilegios. Para ello se debe ingresar a la solapa “Privileges” y agregar un nuevo usuario, haciendo click en “Add User”.



Se deben completar los siguientes campos:

- User name: nombre de usuario
 - En este caso: tpji
- Host: desde donde se conectará el usuario
 - En este caso: % (desde cualquier IP)
- Password: contraseña del usuario
 - En este caso: “tpji2015” (sin comillas)
- Re-type: poner de nuevo la contraseña



Server: localhost » Database: tp2

Structure SQL Search Query Export Import Operations Privileges Routines More

Add user

Login Information

User name: Use text field: tpji

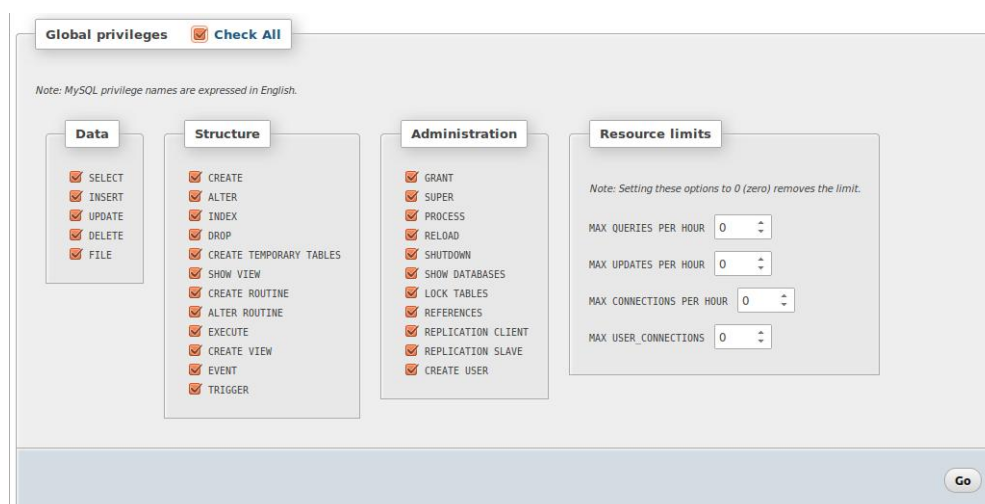
Host: Any host: %

Password: Use text field:

Re-type:

Generate password: Generate

Siguiendo con la página, se puede asignar los privilegios que va a tener este usuario en la base de datos.



Global privileges Check All

Note: MySQL privilege names are expressed in English.

Data

- ☒ SELECT
- ☒ INSERT
- ☒ UPDATE
- ☒ DELETE
- ☒ FILE

Structure

- ☒ CREATE
- ☒ ALTER
- ☒ INDEX
- ☒ DROP
- ☒ CREATE TEMPORARY TABLES
- ☒ SHOW VIEW
- ☒ CREATE ROUTINE
- ☒ ALTER ROUTINE
- ☒ EXECUTE
- ☒ CREATE VIEW
- ☒ EVENT
- ☒ TRIGGER

Administration

- ☒ GRANT
- ☒ SUPER
- ☒ PROCESS
- ☒ RELOAD
- ☒ SHUTDOWN
- ☒ SHOW DATABASES
- ☒ LOCK TABLES
- ☒ REFERENCES
- ☒ REPLICATION CLIENT
- ☒ REPLICATION SLAVE
- ☒ CREATE USER

Resource limits

Note: Setting these options to 0 (zero) removes the limit.

MAX QUERIES PER HOUR 0

MAX UPDATES PER HOUR 0

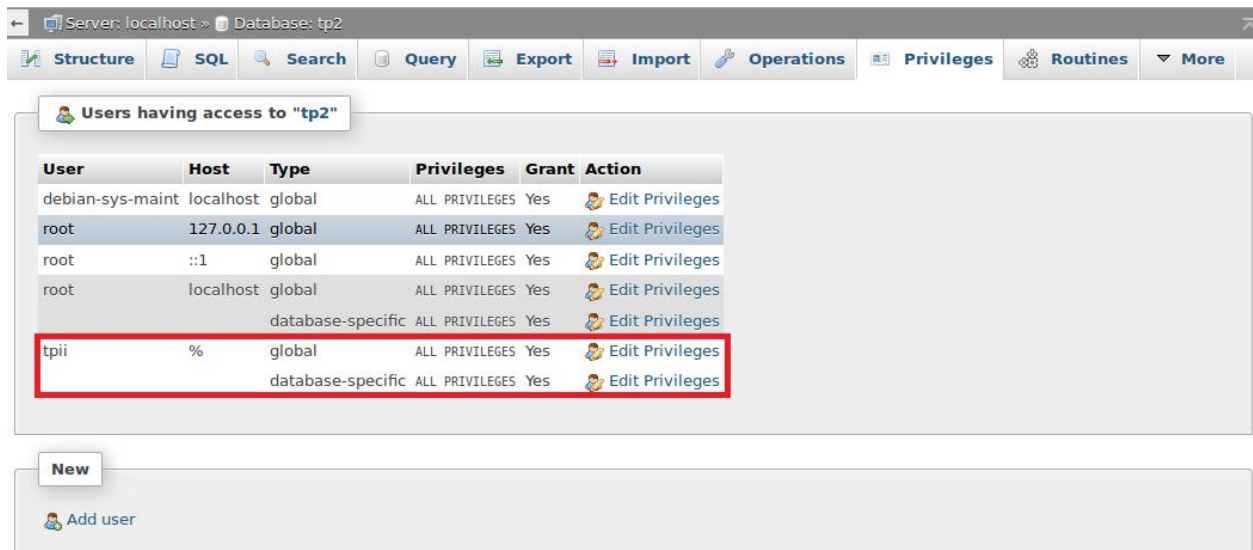
MAX CONNECTIONS PER HOUR 0

MAX USER CONNECTIONS 0

Go

Para completar el registro, hacer click en “Go”.

Finalmente, en la pestaña de “Privileges” se podrá observar el nuevo usuario creado con los privilegios establecidos.



Sistema WEB

El servidor web en el cual se levantarán las páginas, se creó de la siguiente manera:

```
var express = require("express");
var app = express();

var port = process.env.PORT || 5000;
app.listen(port, function() {
  console.log("Listening on " + port);
});
```

Todas las peticiones serán vistas a través del puerto 5000. La variable “app” representa la aplicación WEB que capturara los requerimientos de los diferentes formularios.

Se realizó la configuración del “body-parser” para que se puedan enviar parámetros a través de los métodos “post” de las páginas web.

```
var bodyParser = require('body-parser');
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({
  extend:true
}));
```

Se configuró como motor de vistas a la herramienta “ejs”:

```
app.set('view options', { layout: false });
app.set('view engine', 'ejs');
```

Páginas WEB

El sistema cuenta con cuatro páginas web:

1. inicio.ejs

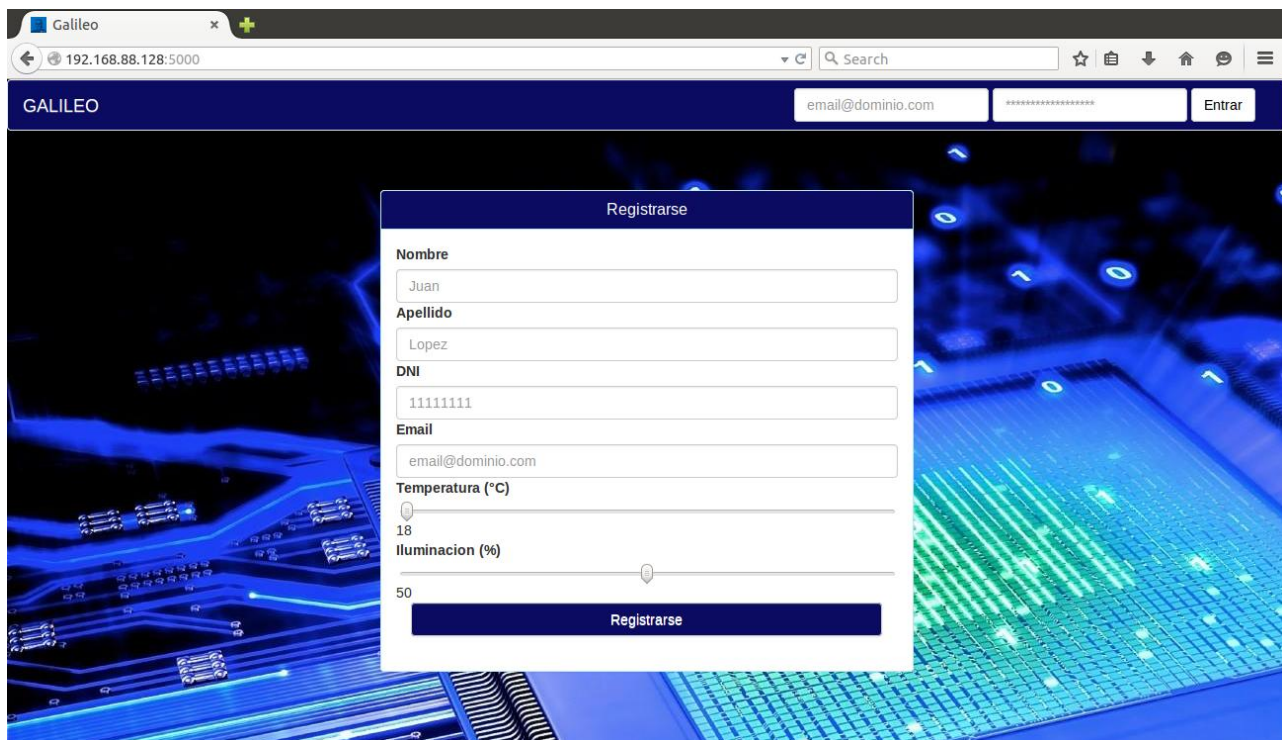
Esta página cuenta con un formulario de registro, el cual permite a un usuario registrarse al sistema y en caso de ya estar registrado, iniciar sesión.

El formulario de registro, permite establecer además de los datos personales, el valor de temperatura y luz con el que le gustaría que se acondicione el ambiente. Una vez registrado, se le enviará por mail la clave ÚNICA de acceso al sistema. Esta clave la podrá utilizar para ingresar al sistema web y modificar su perfil o parámetros, y para ingresar a la habitación, ingresándola a través de un teclado numérico.

Esta página mostrará tres mensajes de error.

- Indicación de registro exitoso
- Indicación de fallo de registro
- Indicación de error de usuario/contraseña

La página se verá de la siguiente manera:



The screenshot shows a web browser window with the address bar displaying '192.168.88.128:5000'. The page title is 'GALILEO'. The main content area features a registration form titled 'Registrarse' with the following fields and controls:

- Nombre:** Text input field containing 'Juan'.
- Apellido:** Text input field containing 'Lopez'.
- DNI:** Text input field containing '11111111'.
- Email:** Text input field containing 'email@dominio.com'.
- Temperatura (°C):** A slider control with a value of 18.
- Iluminacion (%):** A slider control with a value of 50.
- Registrarse:** A dark blue button at the bottom of the form.

The background of the page is a blue and green abstract image with binary code and circuit patterns.

Requerimientos generados al programa “js” desde la página “inicio.ejs”:

- Acceso a la página inicio.ejs a través de la dirección “ip:5000”.

Tal como se mencionó antes, los requerimientos serán capturados a través de la variable “app”. En este caso, se captura el “get” que realiza el navegador para obtener la página web. Al ingresar la método, se le envía la página web sin mensajes de error ya que es la primera petición.

```
app.get("/", function (req, res) {  
  res.render(appDir + '/inicio.ejs', {errorMessage: "", errorMessageRegister: "", successMessageRegister: ""});  
});
```

La variable appDir es la que nos da la dirección donde se encuentra el archivo sobre el cual estamos programando, por lo que sí a las páginas web las ponemos siempre en el mismo directorio, no es necesario configurar la ruta manualmente.

Lo antes mencionado puede realizarse de la siguiente manera:

```
var path = require('path');  
var appDir = path.dirname(require.main.filename);
```

- Registro de usuario

Cuando el usuario presiona el botón “Registrarse”, se genera una petición a través de un método “post” llamada “/registro”. De esta petición, se deben obtener todos los datos que ingreso el usuario. Los mismos, pueden obtenerse a través del parámetro “req.body.parametro”.

En este caso, se debía obtener en primera instancia el email para corroborar que el mismo ya no esté registrado en el sistema, lo mismo ocurriría con el password ya que ambos son únicos. Una vez se haya chequeado esto, el usuario será registrado y se le enviará un email a su casilla con el password generado.

Terminado esto, se responderá al navegador con la página “inicio.ejs”, y un mensaje de “éxito” en caso de que se haya registrado el usuario, o de error en caso contrario.

```

app.post("/registro", function (req, res){
  var regEmail = req.body.email;
  recoveryUserByEmail(regEmail,function (err,content){
    if(err){
      console.log (err);
    }else{
      console.log(content);
      if (content !== null){
        res.render(appDir+'/inicio.ejs',{errorMessage:"",errorMessageRegister:"Usuario ya registrado",successMessageRegister:""});
      }else{
        recoveryAllUsers(function (err,content){
          var passEncontrada = 0;
          while (passEncontrada === 0){
            var regPass = passwordRandom();
            for(var i = 0; i < content.length; i++)
            {
              if(content[i].password !== regPass)
              {
                passEncontrada = 1;
              }
            }
          }
          var regNombre = req.body.nom;
          var regApellido = req.body.ape;
          var regEmail = req.body.email;
          var regDNI = req.body.dni;
          var regTemp = req.body.temp;
          var regLuz = req.body.luz;
          saveUserDataBase(regNombre,regApellido,regDNI,regEmail,regPass,regTemp,regLuz);
          sendEmail(regEmail,regNombre,regPass);
          res.render(appDir+'/inicio.ejs',{errorMessage:"",
            errorMessageRegister:"",
            successMessageRegister:"Verifique su casilla para obtener la contrasena"});
        });
      }
    }
  });
});

```

A continuación, se muestra en forma de ejemplo, el encabezado del “form” que se utilizó en la página “inicio.ejs” para el “/registro”, con la finalidad de visualizar cómo se deben configurar los parámetros “method” y “action”.

```

<form class="form-horizontal" role="form" name="form1" method="post" action="/registro" target="_self" data-toggle="validator">

```

Los mensajes de error que se mencionaron anteriormente, se visualizan en la página de la siguiente manera:

```

<label style="color:red"><%= errorMessageRegister %></label>
<label style="color:green"><%= successMessageRegister %></label>

```

Las variables que están dentro del tag <%= %> son las enviadas cuando se responde la petición al navegador.

- Inicio de sesión

Este requerimiento se recibirá a través de un método “post” llamado “/log” cuando el usuario presione el botón “Entrar”.

En el sistema existirá un “email” y “password” de administrador. Los mismos fueron seteados de la siguiente manera:


```
var emailAdmin = "ppsgalileo@gmail.com";
var claveAdmin = "1234";
```

De la misma manera que en el formulario de registro, los datos ingresados se obtienen a través de `req.body.parametro`.

Esta petición chequeará que el email ingresado este en la base de datos y corresponda con la password ingresada. Luego, verificará si corresponde con el email del administrador, de ser así, redireccionará a la página `"/perfilAdministrador.ejs"`, sino, a la página `"/perfilUsuario.ejs"`. Si el email no corresponde con ningún usuario registrado, se redireccionará nuevamente a la página `"inicio.ejs"` y se enviará un mensaje de error indicando usuario/contraseña inválidos.

```
app.post("/log", function(req, res) {
  recoveryUser(req.body.email, req.body.pass, function (err, content){
    if (err){
      console.log(err);
    }else{
      if (content !== null){
        if (req.body.email !== emailAdmin){
          var dBemail = content[0].email;
          var dBnombre = content[0].nombre;
          var dBapellido = content[0].apellido;
          var dBdni = content[0].dni;
          var dBtemp = content[0].temp;
          var dBluz = content[0].luz;
          var dBpass = content[0].password;
          res.render(appDir+"/perfilUsuario.ejs", {userName: dBnombre,
            userSurname: dBapellido,
            userDNI: dBdni,
            userEmail: dBemail,
            userTemp: dBtemp,
            userLuz: dBluz,
            userPass: dBpass,
            errorMessageEmail: ""});
        }else{
          var dBemail = content[0].email;
          var dBnombre = content[0].nombre;
          var dBapellido = content[0].apellido;
          var dBdni = content[0].dni;
          var dBtemp = content[0].temp;
          var dBluz = content[0].luz;
          var dBpass = content[0].password;
          res.render(appDir+"/perfilAdministrador.ejs", {userName: dBnombre,
            userSurname: dBapellido,
            userDNI: dBdni,
            userEmail: dBemail,
            userTemp: dBtemp,
            userLuz: dBluz,
            userPass: dBpass,
            errorMessageEmail: ""});
        }
      }else{
        res.render(appDir+"/inicio.ejs", {errorMessage: "Usuario/Contraseña invalida",
          errorMessageRegister: "", successMessageRegister: ""});
      }
    }
  });
});
```

2. perfilUsuario.ejs

El perfil de usuario, le mostrará a un usuario registrado todos sus datos y le dará la posibilidad de modificarlos. La página es la siguiente:

Tal como se pudo visualizar en el método “post /log” la redirección al perfil de usuario, no solo contiene la página “perfilUsuario.ejs”, sino que además se agregan datos. Los mismo son utilizados para completar cada campo del formulario que posee esta pagina.

```
res.render(appDir+"/perfilUsuario.ejs", {userName:dbName,
userSurname:dBapellido,
userDNI:dBdni,
userEmail:dBemail,
userTemp: dBtemp,
userLuz: dBluz,
userPass: dBpass,
errorMessageEmail:""});
```

A continuación se muestra un ejemplo de uno de los “input” de la página “perfilUsuario.ejs” para que se visualice como se envia el dato “userName”.

```
<input type="text" class="form-control" name="nom" placeholder="Juan" value="<%= userName %>" required="true">
```

Requerimientos generados al programa “js” desde la página “perfilUsuario.ejs”:

- Cerrar sesión

Esta petición es generada al presionar el boton “Cerrar sesion”. Se produce a través de un método “post” llamado “/cerrarSesion”. Tiene como finalidad redireccionar a la página “inicio.ejs”.

```
app.post("/cerrarSesion",function (req,res){
res.render(appDir+'/inicio.ejs',{errorMessage:"",errorMessageRegister:"",successMessageRegister:""});
});
```

- Modificar perfil

Esta petición se genera al presionar el botón “Modificar perfil” y se produce a través de un método “post” llamado “/modPerfil”.

Este método tomará todos los datos del formulario, chequeara que el email ingresado sea el suyo o en caso de haberlo cambiado que no exista en la base de datos. De ser así, modificará el perfil y re direccionará a la página “perfilUsuario.ejs” completando el formulario con los nuevos datos. Si el email ingresado ya está registrado, re direccionará a la página indicando este error.

```
app.post("/modPerfil",function (req,res){
  var regEmail = req.body.email;
  var regNombre = req.body.nom;
  var regApellido = req.body.ape;
  var regTemp = req.body.temp;
  var regLuz = req.body.luz;
  var regPass = req.body.pass;
  var regDNI = req.body.userDNI;
  console.log (regPass);
  recoveryUserByPass(regPass,function (err,content){
    if (err){
      console.log (err);
    }else{
      var dBid = content[0].id; //Como recupero por clave, siempre hace referencia al usuario que esta modificando el perfil
      if (regPass === claveAdmin){
        if (regEmail === content[0].email){ //Quiere decir que el administrador no cambio el email
          updateUserDataBase(regNombre,regApellido,regEmail,regTemp,regLuz,dBid);
          res.render(appDir+"/perfilAdministrador.ejs", {userName:regNombre,
            userSurname:regApellido,
            userDNI:regDNI,
            userEmail:regEmail,
            userTemp: regTemp,
            userLuz: regLuz,
            userPass: regPass,
            errorMessageEmail:""});
        }else{
          recoveryUserByEmail(regEmail,function(err,contentEmail){
            if (err){
              console.log(err);
            }else{
              if (content === null){ // no hay nadie con ese email
                updateUserDataBase(regNombre,regApellido,regEmail,regTemp,regLuz,dBid);
                res.render(appDir+"/perfilAdministrador.ejs", {userName:regNombre,
                  userSurname:regApellido,
                  userDNI:regDNI,
                  userEmail:regEmail,
                  userTemp: regTemp,
                  userLuz: regLuz,
                  userPass: regPass,
                  errorMessageEmail:""});
              }else{ //NO CAMBIO NINGUN CAMPO
                var dBemail = content[0].email;
                var dBnombre = content[0].nombre;
                var dBapellido = content[0].apellido;
                var dBtemp = content[0].temp;
                var dBluz = content[0].luz;
                res.render(appDir+"/perfilAdministrador.ejs", {userName:regNombre,
                  userSurname:regApellido,
                  userDNI:regDNI,
                  userEmail:regEmail,
                  userTemp: regTemp,
                  userLuz: regLuz,
                  userPass: regPass,
                  errorMessageEmail:"El email ya se encuentra registrado"});
              }
            }
          });
        }
      }
    }
  });
});
```

```

}else{
  if (regEmail === content[0].email){ //Quiere decir que el usuario no cambio el email
    updateUserDataBase(regNombre,regApellido,regEmail,regTemp,regLuz,dBid);
    res.render(appDir+"/perfilUsuario.ejs", {userName:regNombre,
      userSurname:regApellido,
      userDNI:regDNI,
      userEmail:regEmail,
      userTemp: regTemp,
      userLuz: regLuz,
      userPass: regPass,
      errorMessageEmail:""});
  }else{
    recoveryUserByEmail(regEmail,function(err,contentEmail){
      if (err){
        console.log(err);
      }else{
        if (content === null){ // no hay nadie con ese email
          updateUserDataBase(regNombre,regApellido,regEmail,regTemp,regLuz,dBid);
          res.render(appDir+"/perfilUsuario.ejs", {userName:regNombre,
            userSurname:regApellido,
            userDNI:regDNI,
            userEmail:regEmail,
            userTemp: regTemp,
            userLuz: regLuz,
            userPass: regPass,
            errorMessageEmail:""});
        }else{ //NO CAMBIO NINGUN CAMPO
          var dBemail = content[0].email;
          var dBnombre = content[0].nombre;
          var dBapellido = content[0].apellido;
          var dBtemp = content[0].temp;
          var dBluz = content[0].luz;
          res.render(appDir+"/perfilUsuario.ejs", {userName:regNombre,
            userSurname:regApellido,
            userDNI:regDNI,
            userEmail:regEmail,
            userTemp: regTemp,
            userLuz: regLuz,
            userPass: regPass,
            errorMessageEmail:"El email ya se encuentra registrado"});
        }
      }
    });
  }
}

```

3. perfilAdministrador.ejs

El perfil administrador se ve de manera muy similar al del usuario, pero en este caso, se agregan dos botones más a la barra superior. Los mismos son “Información” y “Perfil”.

La página se ve de la siguiente manera:

The screenshot displays the 'perfilAdministrador.ejs' page in a web browser. The browser's address bar shows '192.168.0.13:5000/log'. The page header is dark blue with the text 'GALILEO' on the left and two buttons, 'Informacion' and 'Perfil', in the center. A 'Cerrar sesión' button is located in the top right corner. The main content area features a white form titled 'Usuario Galileo' with a dark blue border. The form contains the following fields and values:

Usuario Galileo	
Nombre:	Administrador
Apellido:	Admin
DNI:	12345678
Email:	ppsgalileo@gmail.com
Contraseña:	1234
Temperatura seleccionada:	25
Intensidad de luz:	65
Modificar perfil	

The background of the page is a blue circuit board pattern.

Al igual que sucedió con la página “perfilUsuario.ejs” al redireccionar al perfil de administrador se envían datos y los mismos cumplen la misma función que en el caso anterior, completar el formulario de perfil.

```
res.render(appDir+"/perfilAdministrador.ejs", {userName:dbName,
userSurname:dBapellido,
userDNI:dBdni,
userEmail:dBemail,
userTemp: dBtemp,
userLuz: dBluz,
userPass: dBpass,
errorMessageEmail:""});
```

Requerimientos generados al programa “js” desde la página “perfilAdministrador.ejs”:

- Los requerimientos “Modificar perfil” y “Cerrar sesion” son los mismos que en el caso de “perfilUsuario.ejs”.
- Información

Este requerimiento se produce al presionar el botón “Información”. La petición viene a través de un método “post” llamado “/historico”.

```
app.post("/historico",function(req,res){
  recoveryAllUsers(function(err,content){
    if (err)
      console.log(err)
    else{
      var dataObject = [];
      for (var i=0; i < content.length; i++){
        dataObject.push({ nombre:content[i].nombre,apellido:content[i].apellido,dni:content[i].dni,email:content[i].email});
      }
      recoveryAllAuditoria(function(err,content){
        if (err)
          console.log(err)
        else{
          var dataObjectAud = [];
          var dataObjectAudHistorico = [];
          for (var i=0; i < content.length; i++){
            if(content[i].fechaSalida != null){
              dataObjectAudHistorico.push({email:content[i].email,fechaEntrada:content[i].fechaEntrada,fechaSalida:content[i].fechaSalida});
            }else{
              dataObjectAud.push({email:content[i].email,fechaEntrada:content[i].fechaEntrada});
            }
          }
          console.log();
          res.render(appDir + '/historicoAdministrador.ejs',
            {data:dataObject,dataAuditoriaHistorico:dataObjectAudHistorico,dataAuditoria:dataObjectAud});
        }
      });
    }
  });
});
```

Este método nos redireccionará a una página llamada “historicoAdministrador.ejs”. En esta página se mostrará información de los usuarios registrados en el sistema, los accesos históricos y los usuarios que actualmente están en la habitación.

Para enviar la información, se realizan dos consultas. La primera es para obtener todos los usuarios, y la segunda, para obtener los registros de accesos a la habitación. En el caso de esta última, se discrimina por la fecha de salida (si es NULL) si el registro es histórico o actual, entendiéndose por actual a un usuario que todavía no salió de la habitación. Los datos son enviados a la página WEB en formato “JSON”. En esta página hay un template “ejs” el cual crea las tablas a partir de los datos enviados por el programa.

A continuación, se muestra una porción de la página “historicoAdministrador.ejs” donde se muestra como arma las tablas a partir de los datos que se envían.

```
<%  
  for (var i = 0; i < data.length; i++) {  
    %>  
    <tbody>  
    <tr>  
      <td><%= data[i].nombre %></td>  
      <td><%= data[i].apellido %></td>  
      <td><%= data[i].dni %></td>  
      <td><%= data[i].email %></td>  
    </tr>  
    </tbody>  
  } %>
```

- Perfil

Esta petición es generada al presionar el botón perfil, y llega a través de un método “post” llamado “/perfil”. Esto tiene como función regresar al perfil del administrador, en caso de que se encuentre en la página “historicoAdministrador.ejs”.

```
app.post("/perfil",function(req,res){  
  recoveryUser(emailAdmin,claveAdmin,function (err,content){  
    if (err){  
      console.log(err);  
    }else{  
      if (content !== null){  
        var dBemail = content[0].email;  
        var dBnombre = content[0].nombre;  
        var dBapellido = content[0].apellido;  
        var dBdni = content[0].dni;  
        var dBtemp = content[0].temp;  
        var dBluz = content[0].luz;  
        var dBpass = content[0].password;  
        res.render(appDir+"/perfilAdministrador.ejs", {userName:dBnombre,  
                                                         userSurname:dBapellido,  
                                                         userDNI:dBdni,  
                                                         userEmail:dBemail,  
                                                         userTemp: dBtemp,  
                                                         userLuz: dBluz,  
                                                         userPass: dBpass,  
                                                         errorMessageEmail:""});  
      }else{  
        console.log("Error de acceso en base de datos - app.post(\"/perfil\"....)");  
      }  
    }  
  });  
});
```


4. historicoAdministrador.ejs

Tal como se mencionó en el punto anterior, esta página mostrará en forma de tablas información sobre los usuarios registrados, los accesos históricos y los actuales. La página es la siguiente:

Nombre	Apellido	DNI	Email
Administrador	Admin	12345678	ppsgalileo@gmail.com
Gaston	Bezzi	12345687	gastonbezzi@gmail.com
Milton	Meroni	35798492	miltonmeroni@gmail.com
Gaston	Maron	36996998	gastonmaron@gmail.com

Email	Fecha entrada	Fecha salida
miltonmeroni@gmail.com	15:41 - 13/10/2015	15:41 - 13/10/2015
gastonmaron@gmail.com	15:57 - 13/10/2015	15:08 - 13/10/2015
ppsgalileo@gmail.com	16:13 - 13/10/2015	15:49 - 13/10/2015
miltonmeroni@gmail.com	15:49 - 13/10/2015	15:49 - 13/10/2015

Email	Fecha entrada
-------	---------------

Los requerimientos generados al programa “js” desde la página “historicoAdministrador.ejs” son “Cerrar sesion”, “Información” y “Perfil”. Estos son los mismos que se explicaron en los puntos anteriores.

Envío de emails

Para enviar emails a través de nodejs, tal como se explicó en el comienzo del informe, es necesario la instalación de la librería “emailjs”. A continuación, se muestra la configuración del servidor de email.

```
var email = require("emailjs");
var serverEmail = email.server.connect({
  user: "ppsgalileo@gmail.com",
  password: "ingenieriaencomputacion",
  host: "smtp.gmail.com",
  ssl: true
});
```

Se creó una función que recibe el email del destinatario, el nombre y la password que deberá utilizar para ingresar al sistema. Esta función envía al usuario la clave ÚNICA de acceso.

```
function sendEmail(email,nombre,pass){
  serverEmail.send({
    text: "Sr/a. "+nombre+" \n \nSu clave de acceso es: "+pass+"\n \nSaludos,\n \nGalileo",
    from: "Galileo <ppsgalileo@gmail.com>",
    to: nombre+" "+email,
    //cc: "else <else@your-email.com>",
    subject: "Clave de acceso - Galileo"
  }, function(err, message) { if (err!=null) console.log(err); });
}
```

El mensaje que se recibe es el siguiente:



Uso del microcontrolador y uso de sensores y actuadores

Para tarea del microcontrolador, se tuvo que combinar la programación en lenguaje C y en el Node.js. Esto se debe a que el programa en lenguaje C captura los eventos del teclado y los escribe en un archivo, mientras que el programa principal del Node.js es una función que se ejecutará cada un segundo, leyendo ese archivo y verificando que la contraseña sea la de algún usuario.

Programa en lenguaje C

Este programa captura los eventos provenientes del teclado y los pone en un archivo de texto. Este archivo se debe encontrar en el mismo directorio que la aplicación. Para que pueda detectar el teclado, hay que averiguar a qué evento dentro del directorio “/dev/input/” ya que el sistema operativo tiene varios eventos que atender. Para ello antes de ejecutar el programa se debe probar en una consola cual es el evento buscado:


```

var exec = require('child_process').exec;
var child;
var evento = "/dev/input/event1";
var ejecutarTeclas = "cd "+appDir+"; ./teclado "+evento;
child = exec(ejecutarTeclas, function (error, stdout, stderr) {
  console.log('stdout: ' + stdout);
  console.log('stderr: ' + stderr);
  if (error !== null) {
    console.log('exec error: ' + error);
  }
});

```

El programa principal va abrir el archivo “teclas.txt” en modo lectura, y verificar si se presionó la tecla “Enter”.

En caso de que se haya presionado, se obtendrá la contraseña del archivo y se limpiara el mismo. Una vez realizado esto, se obtendrá el usuario de la base de datos. Cuando se ingresa esta nueva clave, en caso de que el usuario exista, pueden haber ocurrido dos situaciones. Que el usuario esté ingresando a la habitación, o que el mismo esté saliendo. Por lo tanto, al obtener al usuario, se debe chequear que este no tenga una auditoría activa, si esto NO es así, se debe verificar que no haya ningún otro usuario en la habitación. De ser así, se creará una nueva auditoría para registrar el ingreso y se da inicio a la función “simulacionSensores”. Esta función recibe los parámetros de temperatura y luz que el usuario predefinio en su perfil, y tiene como objetivo simular el acondicionamiento de la habitación.

Si existe una auditoría activa por este usuario, se actualiza la misma con la fecha de salida, y se detiene la función de acondicionamiento.

Como primera medida, la función “simulacionSensores” simula el encendido de la luz al porcentaje indicado por el parámetro “luz”. Luego, dentro de sí misma, tiene una sección que se repetirá cada un segundo con la finalidad de simular el acondicionamiento de la temperatura del ambiente. En esta sección se chequeara si se debe bajar, aumentar, o mantener la temperatura. La temperatura actual (sensor) se generó con una función que aporta valores aleatorios entre 18 y 30 grados.

Como se explicó anteriormente, los sensores serán simulados en la función “simulacionSensores”. Los mismos serán un sensor de luz y un sensor de temperatura. El display será simulado con el uso de la terminal.

Posibles errores

Error de permiso denegado de ejecución de ./teclado:

```
ubuntu@ubuntu:~/practica2$ nodejs express.js
body-parser deprecated undefined extended: provide extended option express.js:8:2
0
Listening on 5000
stdout:
stderr: /bin/sh: 1: ./teclado: Permission denied
exec error: Error: Command failed: /bin/sh: 1: ./teclado: Permission denied
```

Solución:

Dentro del directorio, cambiar los permisos del programa ejecutable con el siguiente comando:

- *chmod 775 teclado*

Error de permiso denegado del evtest:

```
ubuntu@ubuntu:~/practica2$ nodejs express.js
body-parser deprecated undefined extended: provide extended option express.js:8:2
0
Listening on 5000
stdout: /home/ubuntu/practica2/teclas.txt
stderr: evtest: Permission denied
exec error: Error: Command failed: evtest: Permission denied
```

Solución:

- Ejecutar el programa como administrador anteponiendo el comando *sudo*
 - *sudo nodejs express.js*

Comunicación con la PC

Para la comunicación con la PC (que en nuestro caso es la base de datos) se implementaron una serie de variables, que representan los parámetros para la conexión con la base de datos; y una serie de métodos que establecen la conexión y realizan consultas en la base de datos.

Las variables son las siguientes:

- `var ipDataBase = '192.168.88.128';` // ip de la base de datos
- `var usrDataBase = 'tpii';` // nombre de usuario
- `var passDataBase = 'tpii2015';` // contraseña
- `var nameDataBase = 'tp2';` // nombre de la base de datos

Estas variables van a ser configuradas cada vez que se lo necesite, en caso de que se haya cambiado la IP de la base de datos o si, cuando se realizó la importación de la base de datos, el usuario y/o la contraseña que se establecieron sean diferentes.

Los métodos son sobre las entidades de la base de datos, Usuario y Auditoria. Cabe destacar que en todos los métodos, se configura los parámetros para la conexión con la base de datos, se efectúa la conexión, se realiza la consulta y, al finalizar el método, se cierra la conexión.

Sobre la entidad Usuario, se establecieron los siguientes métodos:

recoveryAllUsers(callback)

- Recupera todos los usuarios que existen en la base de datos. En caso de que no haya ninguno retorna (NULL, NULL). En caso de error, devuelve el error producido.

```
function recoveryAllUsers(callback){
    var mysql = require('mysql');
    var connection = mysql.createConnection({
        host : ipDataBase,
        user : usrDataBase,
        password : passDataBase,
        database : nameDataBase
    });
    connection.connect();

    var query = 'SELECT * FROM usuario u';

    var resQuery = connection.query(query,function(err, rows, fields) {
        if (!err){
            if (rows.length > 0)
                return callback(null, rows);
            else
                return callback (null,null)
        }else{
            return callback (err,null);
        }
    });
    connection.end();
}
```

recoveryUser(email,pass,callback)

- Consulta si existe el email y el password de un usuario en particular. Retorna el mismo resultado que el método recoveryAllUser.

```
function recoveryUser(email,pass,callback){
    var mysql = require('mysql');
    var connection = mysql.createConnection({
        host : ipDataBase,
        user : usrDataBase,
        password : passDataBase,
        database : nameDataBase
    });
    connection.connect();

    var query = 'SELECT * FROM usuario u WHERE (u.email="'+email+'" and u.password="'+pass+'")';

    var resQuery = connection.query(query,function(err, rows, fields) {
        if (!err){
            if (rows.length > 0)
                return callback(null, rows);
            else
                return callback (null,null)
        }else{
            return callback (err,null);
        }
    });
    connection.end();
}
```


recoveryUserByEmail(email,callback)

- Recupera el usuario que tenga el email enviado por parámetro. Retorna el mismo resultado que el método recoveryAllUser.

```
function recoveryUserByEmail (email,callback){
  var mysql      = require('mysql');
  var connection = mysql.createConnection({
    host      : ipDataBase,
    user      : usrDataBase,
    password  : passDataBase,
    database  : nameDataBase
  });
  connection.connect();

  var query = 'SELECT * FROM usuario u WHERE u.email="'+email+'"';

  var resQuery = connection.query(query,function(err, rows, fields) {
    if (!err){
      if (rows.length > 0)
        return callback(null, rows);
      else
        return callback (null,null)
    }else{
      return callback (err,null);
    }
  });
  connection.end();
}
```

recoveryUserByPass(pass,callback)

- Recupera el usuario que tenga la contraseña enviada por parámetro. Retorna el mismo resultado que el método recoveryAllUser

```
function recoveryUserByPass (pass,callback){
  var mysql      = require('mysql');
  var connection = mysql.createConnection({
    host      : ipDataBase,
    user      : usrDataBase,
    password  : passDataBase,
    database  : nameDataBase
  });
  connection.connect();

  var query = 'SELECT * FROM usuario u WHERE u.password="'+pass+'"';

  var resQuery = connection.query(query,function(err, rows, fields) {
    if (!err){
      if (rows.length > 0)
        return callback(null, rows);
      else
        return callback (null,null)
    }else{
      return callback (err,null);
    }
  });
  connection.end();
}
```

saveUserDataBase(nombre,apellido,dni,email,pass,temp,luz)

- Guarda un usuario con todos los valores que vienen por parámetro.

```
function saveUserDataBase(nombre,apellido,dni,email,pass,temp,luz){
  var mysql      = require('mysql');
  var connection = mysql.createConnection({
    host      : ipDataBase,
    user      : usrDataBase,
    password  : passDataBase,
    database  : nameDataBase
  });
  connection.connect();

  var valuesInsert = {nombre: nombre, apellido: apellido, dni: dni, email: email, password: pass, temp:temp, luz:luz};
  var query = connection.query('INSERT INTO usuario SET ?', valuesInsert, function(err, result) {
    if (err)
      console.log(err);
  });
  connection.end();
}
```

updateUserDataBase(nombre,apellido,email,temp,luz,id)

- Actualiza los atributos que el usuario puede llegar a modificar. Entre estos se encuentran el nombre, apellido, email, temperatura y luz.

```
function updateUserDataBase(nombre,apellido,email,temp,luz,id){
  var mysql = require('mysql');
  var connection = mysql.createConnection({
    host : ipDataBase,
    user : usrDataBase,
    password : passDataBase,
    database : nameDataBase
  });
  connection.connect();

  var valuesUpdate = {nombre: nombre, apellido: apellido, email: email, temp:temp, luz:luz};
  var querySentence = 'UPDATE usuario SET ? WHERE id='+id+'';
  var query = connection.query(querySentence,valuesUpdate, function(err, result) {
    if (err){
      console.log(err);
    }
  });
  connection.end();
}
```

Sobre la entidad auditoria se realizaron los siguientes métodos:

recoveryAllAuditoria(callback)

- Recupera todas las auditorias que se encuentran en la base de datos. Retorna las filas obtenidas o NULL en caso de que no haya. En caso de error, envía el error que se ocasionó.

```
function recoveryAllAuditoria (callback){
  var mysql = require('mysql');
  var connection = mysql.createConnection({
    host : ipDataBase,
    user : usrDataBase,
    password : passDataBase,
    database : nameDataBase
  });
  connection.connect();

  var query = 'SELECT * FROM auditoria a';

  var resQuery = connection.query(query,function(err, rows, fields) {
    if (!err){
      if (rows.length > 0)
        return callback(null, rows);
      else
        return callback (null,null)
    }else{
      return callback (err,null);
    }
  });
  connection.end();
}
```

saveAuditoriaDataBase (email)

- Genera la fecha de entrada con el formato “hh:min – dd/mm/yy” e inserta dicha fecha junto con el email que viene por parámetro.

```
function saveAuditoriaDataBase (email){
    var hoy = new Date();
    var dd = hoy.getDate();
    var mm = hoy.getMonth()+1; //Enero es el mes 0
    var yy = hoy.getFullYear();
    var hh = hoy.getHours();
    var min = hoy.getMinutes();
    if (parseInt(min,10) < 10){
        min = '0'+min;
    }
    var fechaEntrada = hh+":"+min+" - "+dd+"/"+mm+"/"+yy;

    var mysql = require('mysql');
    var connection = mysql.createConnection({
        host      : ipDataBase,
        user       : usrDataBase,
        password   : passDataBase,
        database   : nameDataBase
    });
    connection.connect();

    var valuesInsert = {fechaEntrada: fechaEntrada, email: email};
    var query = connection.query('INSERT INTO auditoria SET ?', valuesInsert, function(err, result) {
        if (err)
            console.log(err);
    });

    connection.end();
}
```

updateAuditoriaDataBase(email)

- Genera la actualización de la auditoria, es decir, le asigna una fecha a la fecha de salida que esté establecida como NULL en una auditoria que contenta el email que viene por parámetro.

```
function updateAuditoriaDataBase(email){
    var hoy = new Date();
    var dd = hoy.getDate();
    var mm = hoy.getMonth()+1; //Enero es el mes 0
    var yy = hoy.getFullYear();
    var hh = hoy.getHours();
    var min = hoy.getMinutes();
    if (parseInt(min,10) < 10){
        min = '0'+min;
    }

    var fechaSalida = hh+":"+min+" - "+dd+"/"+mm+"/"+yy;

    var mysql = require('mysql');
    var connection = mysql.createConnection({
        host      : ipDataBase,
        user       : usrDataBase,
        password   : passDataBase,
        database   : nameDataBase
    });
    connection.connect();
    var valuesInsert = {fechaSalida: fechaSalida};
    var query = connection.query('UPDATE auditoria a SET ? WHERE (a.fechaSalida IS NULL)&&(a.email="'+email+'");', valuesInsert,
function(err, result) {
    if (err)
        console.log(err);

});

    connection.end();
}
```

Posibles errores

Conexión rechazada de la base de datos

```
ubuntu@ubuntu:~/practica2$ sudo nodejs express.js
body-parser deprecated undefined extended: provide extended option express.js:8:
20
Listening on 5000
1234
password: 1234
{ [Error: connect ECONNREFUSED]
  code: 'ECONNREFUSED',
  errno: 'ECONNREFUSED',
  syscall: 'connect',
  fatal: true }
```

Solución

Esto puede ser por la mala configuración del servidor mysql que contiene la base de datos. Una configuración que puede funcionar es la que se encuentra en el archivo “my.cnf” en la carpeta “database”. Para poder copiar este archivo en la carpeta correspondiente (/etc/mysql/) se debe ejecutar los siguientes comandos:

- `cd database`
- `cp my.cnf /etc/mysql`
 - en caso de no poder efectuar este comando debido al error “Permission denied”, anteponer el comando *sudo*.

```
^Cubuntu@ubuntu:~/practica2$ cd database/
ubuntu@ubuntu:~/practica2/database$ ls
my.cnf  tp2.sql.zip
ubuntu@ubuntu:~/practica2/database$ cp my.cnf /etc/mysql/
cp: cannot create regular file '/etc/mysql/my.cnf': Permission denied
ubuntu@ubuntu:~/practica2/database$ sudo cp my.cnf /etc/mysql/
ubuntu@ubuntu:~/practica2/database$ cd ..
ubuntu@ubuntu:~/practica2$ sudo nodejs express.js
body-parser deprecated undefined extended: provide extended option express.js:8:
20
Listening on 5000
1234
password: 1234
{ [Error: connect ECONNREFUSED]
  code: 'ECONNREFUSED',
  errno: 'ECONNREFUSED',
  syscall: 'connect',
  fatal: true }
```

Si se ejecuta de nuevo y sigue arrojando el mismo error, resetear el servicio del servidor mysql:

- `sudo service mysql restart`

```

ubuntu@ubuntu:~/practica2$ sudo service mysql restart
ubuntu@ubuntu:~/practica2$ sudo nodejs express.js
body-parser deprecated undefined extended: provide extended option express.js:8:
20
Listening on 5000
1234
password: 1234
Usuario: ppsgalileo@gmail.com
DISFRUTE SU ESTADIA
Ambiente a acondicionar -> Temperatura: 25 Luz:65
Enciendo luz hasta 65 %
Temperatura sensada al momento del ingreso (int): 29
Temperatura pedida (int): 25
-->Bajar temperatura 29

```

Volver a ejecutar y el problema estará resuelto.

Diferencias entre sistema real y simulado

En la simulación hay varias cosas que no se van a poder implementar en el sistema real o que al menos ahora no se pueden determinar con exactitud. Por ejemplo, la temperatura del sensor se hace aumentando o disminuyendo de a un grado por iteración, siendo que se va a tener variables como el tiempo en que el sensor entrega la muestra, realizar la conversión de tensión a grados y, luego hacer las comparaciones.

Otro factor delimitante es que el display no va a poder mostrar todos los mensajes que aparecerán en la terminal, sino que, al ser de ocho dígitos por fila, el mensaje va a ser acotado.

Concurrencia

En primera medida, se han encontrado algunos problemas. Las pruebas que se realizaron fueron:

- Registro de dos personas con diferente mail al mismo tiempo.
- Registro de dos personas con el MISMO mail al mismo tiempo.
 - El sistema aceptó una solicitud, realizo las acciones correspondientes y al otro usuario le advirtió que el mail ya existía.
- Acceso al sistema con el mismo usuario.
- Acceso al sistema con distintos usuarios.
- Modificación de perfil con distintos usuarios.
- Modificación de perfil con el mismo usuario.

- Si se quiere modificar el perfil en dos computadoras diferentes, la base de datos va a tomar los datos del último que hizo click en “modificar perfil”, haciendo que el que primero hizo click no vea los cambios reflejados de la modificación.