ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

SISTEMA DE RIEGO PARA INVERNADEROS CON INTERFACE WEB UTILIZANDO UN ARDUINO YUM

ALMACENAMIENTO DE DATOS EN LA NUBE

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

JONATHAN ANDRÉS ALVAREZ IMBAQUINGO

jonathan.alvarez@epn.edu.ec

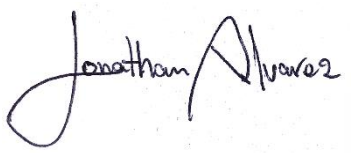
RAMIRO EDUARDO MOREJON TOBAR

ramiro.morejon@epn.edu.ec

DMQ, FEBRERO 2023

CERTIFICACIONES

Yo, JONATHAN ANDRES ALVAREZ IMBAQUINGO declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.



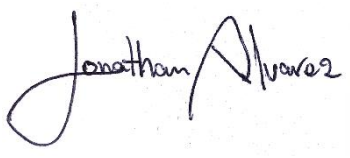
JONATHAN ANDRES ALVAREZ IMBAQUINGO

Certifico que el presente trabajo de integración curricular fue desarrollado por JONATHAN ANDRES ALVAREZ IMBAQUINGO, bajo mi supervisión.

Ing. RAMIRO EDUARDO MOREJON TOBAR, M.Sc.
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.



JONATHAN ANDRES ALVAREZ IMBAQUINGO

Ing. RAMIRO EDUARDO MOREJÓN TOBAR, M.Sc.

DEDICATORIA

A mis queridos padres Xavier y Mónica y mis hermanos Brandon y Solange, quienes son los pilares que han permitido mi viaje académico. Su amor incondicional e innumerables sacrificios han sido la fuerza que me ha llevado a perseverar ante las dificultades para conseguir esta meta. Les dedico este trabajo, con mucha gratitud y cariño.

AGRADECIMIENTO

Quiero expresar mi profunda gratitud a todos los que me han brindado su apoyo incondicional durante mi trayectoria académica.

En primer lugar, quiero dar las gracias al Ing. Ramiro Morejón, M. Sc, mi director de tesis, por su sabiduría, paciencia y guía constante. Su experiencia y aliento han sido fundamentales para la realización de este trabajo.

También quiero agradecer al Dr. Hernán Barba, mi tutor, por su apoyo y estímulo a lo largo de toda mi carrera académica.

A mi madre Mónica, por su amor incondicional, su fuerza y su soporte inquebrantable. A mi padre Xavier, por su trabajo incansable, su amor y su aliento. A ambos, por estar siempre a mi lado, por guiarme y por ser mi pilar en los momentos difíciles. A mi hermana Solange por su inspiración, cariño y ser la fuerza que me lleva a esforzarme aún más. A mi hermano Brandon por su coraje, fortaleza y soporte. A todos por su constante presencia y compañía en los momentos más difíciles.

A mis abuelitos y abuelitas, siempre estaré agradecido por su inmenso cariño.

A mis amigos Ace, Diego, Jhon, Joshua, Lenin, Suas, Stalin y Zeus por los buenos momentos compartidos. A Jonathan y Jossue, por su apoyo y aliento durante la carrera, lo cual siempre estaré agradecido.

Finalmente, a la Universidad y a todo su personal, por las oportunidades y el apoyo brindado para mi crecimiento académico.

ÍNDICE DE CONTENIDO

RESUMEN

El presente trabajo tiene como objetivo presentar una solución para el monitoreo de variables ambientales de un sistema de invernaderos mediante la creación de una aplicación web y el almacenamiento de la información obtenida en una base de datos en la nube. Los invernaderos tienen un rol crucial en la agricultura moderna ya que proveen un ambiente controlado que permite el crecimiento de una amplia variedad de cultivos. El uso de servicios en la nube brinda una solución moderna, segura y escalable lo cual permite desarrollar aplicaciones de acorde a las necesidades del invernadero. Este trabajo se enfoca en las características que ofrecen las principales plataformas de servicios en la nube, los pasos necesarios para desarrollar la aplicación web que será desplegada en uno de estos servicios, el despliegue de la aplicación y la base de datos en la nube, y el método de comunicación entre el sistema de invernaderos con la herramienta web creada. Como resultado del trabajo se tendrá una aplicación web en la nube que podrá ser accedida a través de dispositivos que tengan conexión a internet, además una base datos que almacenará los datos históricos de las variables ambientales provenientes del sistema de invernaderos.

PALABRAS CLAVE: nube, despliegue, plataforma, servicio, Python, Azure, desarrollo, invernadero, almacenar, variables ambientales.

ABSTRACT

The present work aims to present a solution for monitoring environmental variables of a greenhouse system through the creation of a web application and storage of their data in a cloud-based database. Greenhouses have a crucial role in modern agriculture as they provide a controlled environment that allows for the growth of a wide variety of crops. The use of cloud services provides a modern, secure and scalable solution that allows for the development of applications according to the needs of the greenhouse. This work focuses on the features offered by main cloud platforms, which steps are necessary to develop the web application that will be deployed in one of these services, the deployment of the application and the database in a cloud service provider, and the method of communication between the greenhouse system and the web app created. As a result of the work, a cloud-based web application will be obtained that could be accessed through devices which have internet connection, also a database that will store historical data of the environmental variables provided by the greenhouse system.

KEYWORDS: cloud, deployment, platform, service, Python, Azure, development, greenhouse, storage, environmental variables.

1 INTRODUCCIÓN

Un invernadero es un sistema que posibilita modificar y adecuar el ambiente que permite a las plantas crecer en climas o en temporales donde no es adecuado su desarrollo. Los invernaderos se utilizan comúnmente para cultivar frutas, vegetales y flores, permitiendo un ambiente controlado para cada tipo de planta. Dicho ambiente puede tener condiciones climáticas que pueden ir desde climas templados a temperaturas similares a las de un desierto[1]. Para el control tradicional de las variables climáticas los invernaderos cuentan con propiedades en su diseño que favorecen ciertas temperaturas a otras. Elementos como actuadores y sensores permiten mayor control sobre las variables ambientales, esto hace su supervisión tenga gran importancia por lo cual en este trabajo se presentará una solución para el monitoreo de dichas variables usando servicios en la nube [2].

Los invernaderos permiten que las cosechas sean más eficientes, sostenibles, resistentes, además en nuestro caso presentará innovación [3]. Estas propiedades fomentan a que exista mayor cantidad y mejor calidad de productos agrícolas, promoviendo acercarse a la seguridad alimentaria. Expertos en el tema indican que no existe una única solución que provea seguridad alimentara en un futuro. Para alcanzar fuentes sustentables de alimento se debe atacar el problema desde varios ángulos y uno de ellos es el uso de la tecnología para optimizar la producción[4]. El enfoque tecnológico en la agricultura permite implementar sistemas de cultivo en sitios no tradicionales como invernaderos y centros de producción urbanos, lo que ofrece la posibilidad de cultivar en lugares diferentes fuera del campo.

El desarrollo de la tecnología de transmisión de datos ha permitido el surgimiento de dispositivos capaces de enviar información sin que sea necesaria la intervención de alguna persona [5]. Además, avances en el diseño de invernaderos ha posibilitado que las plantaciones puedan llevarse de mejor manera en sitios urbanos e incrementar ciclos de siembra. Las consecuencias de esto se pueden observar en reducción de costos al no tener que desplazar al operario al campo, minimizando la intervención humana gracias al monitoreo remoto.

Mayor monitoreo y adecuación de variables ambientales en invernaderos ha contribuido a que aumente la sostenibilidad de la actividad agrícola al optimizar recursos como el agua, fertilizantes y la demanda energética. El balance entre sostenibilidad y adecuada producción debe tomarse en cuenta debido a que el crecimiento propicio de las plantas depende principalmente de parámetros ambientales como son: luminosidad, humedad ambiental, humedad de suelo y niveles de PH del agua que son variables que presentan

gran dificultad de control en un entorno abierto [6]. Para esto los sistemas de invernadero son una gran alternativa que permite disminuir el consumo de estos recursos produciendo aumento en las ganancias y que no requiera la presencia constante de una persona a su cuidado.

Para tener mayor flexibilidad se ha visto en la necesidad de implementar mediante servicios en la nube sistemas que permitan presentar y almacenar los datos provenientes del invernadero. Este tipo de sistemas permiten a personas quienes de otra manera requerirían acudir al invernadero para su monitoreo lo realicen desde cualquier sitio mediante dispositivos que tengan acceso a internet. Una implementación de este tipo permite el monitoreo, análisis, y gestión del invernadero desde posibles grandes distancias[7].

El sistema propuesto enviará datos entregados del sistema de invernaderos en tiempo real a una plataforma en la nube donde serán guardados y presentados. Los datos podrán ser observados de forma histórica mediante tablas y gráficos en una interfaz web amigable para el usuario. La información podrá ser analizada o procesada ya que se contará con ella en una base de datos localizada en la nube.

1.1 OBJETIVO GENERAL

Implementar por medio de plataformas de servicios en la nube el almacenamiento de datos provenientes de un sistema de invernaderos y su presentación a través de una aplicación web que pueda ser accedida mediante internet.

1.2 OBJETIVOS ESPECÍFICOS

1. Seleccionar la plataforma de servicios en la nube y familiarización con la misma.
2. Definir una estructura de variables obtenidas del sistema de riego para su almacenamiento en una base de datos, procesamiento y presentación.
3. Determinar el lenguaje de programación en el cual se desarrollará la aplicación web y proceder con su desarrollo.
4. Establecer la comunicación entre el módulo localizado en el invernadero y el servicio en la nube desplegado.
5. Visualizar los datos históricos almacenados en la plataforma a través de la aplicación web.

1.3 ALCANCE

En una primera fase se realizará una investigación teórica acerca de los diferentes servicios que ofrecen almacenamiento en la nube, transmisión de los datos provenientes del invernadero e implementación de una página web. En este aparatado presentará diferentes implementaciones con características similares al proyecto, así será posible encontrar una solución adecuada para la dimensión del proyecto.

Posteriormente, usando el servicio en la nube que fue seleccionado en la primera fase a través de internet se receptará y almacenará en una base de datos la información proveniente del sistema de riego. La información será procesada para que pueda ser accedida en forma de variables con un valor en el tiempo para que puedan ser presentadas apropiadamente.

A través de la implementación de una página web será posible desplegar gráficas y tablas que permitirán observar las variables seleccionadas en función del tiempo. De igual manera será posible acceder al historial de las variables para tener una idea de cómo se ha comportado el sistema.

1.4 MARCO TEÓRICO

Los invernaderos se definen como estructuras diseñadas para controlar los parámetros más importantes que afectan el crecimiento de los cultivos, como son: luminosidad solar, humedad relativa, temperatura, intensidad lumínica, todo esto para incrementar las cosechas sin importar la temporada. Su objetivo principal es ofrecer un ambiente que sea adecuado para producción intensiva de plantaciones a la vez que se los protege de pestes y condiciones climáticas adversas.

1.4.1 Características del Invernadero

Un invernadero es una estructura que emplea la energía de la radiación solar interceptada para crear un microclima más favorable, que sea adecuado para los requerimientos de la plantación. La interceptación de la energía solar permite reducir los costos de operación al apoyar sistemas de actuación que regulen el microclima dentro del invernadero. Se debe tener en cuenta que la optimización de la energía solar recibida depende en gran medida de la geometría, orientación, área de ventilación, sistemas para producir sombra y el material del que está hecho el invernadero. [9]

El invernadero está constituido de un armazón cubierto de plástico o vidrio. Puede ser de diferentes formas y tamaños, además de varios precios. La localización tiene influencia en estilo del a construcción y la cantidad de luz que las plantas recibirán, esto se debe tener en cuenta dependiendo del tipo de planta que se desea cosechar. El tipo de acceso que tendrá la persona a cargo del invernadero también debe ser considerado, en caso de un sistema automatizado en el cual la interacción directa del encargado será minimizada existirá la posibilidad de que la localización del invernadero sea remota lo que puede resultar en una reducción del costo en el terreno y en el mantenimiento.

Al considerar el tipo de estructura del invernadero se debe tener en cuenta si la construcción se realizará cercana a otra construcción, estará aislada o se realizará en un espacio muy reducido. Una estructura independiente proporciona más espacio de crecimiento y se puede colocar en cualquier lugar de la propiedad, invernaderos que son construidos contra una estructura existente en uno o varios lados y son buena opción cuando se desea construir en un área residencial. También se puede considerar cuando algunas plantas son montadas cerca de una ventana que provee poco calor. Es importante tener en cuenta que, si la construcción se realizará en áreas susceptibles a vientos o tormentas, tener un invernadero anclado a una base con una estructura sólida puede evitar que se lo lleve el viento.

La proporción debe ser tomada en cuenta para maximizar la exposición al sol mientras se limita la pérdida de calor, esto se logra al tener estructuras largas y angostas. Depende del espacio disponible pero una relación de ancho a largo de 1:3 es ideal. [10]. Dos factores críticos para el invernadero son materiales y forma. Algunos están completamente hechos de plástico o vidrio, mientras que otros son una combinación de materiales. Algunos tienen techos redondeados, mientras que otros están inclinados. Cada uno tiene ventajas y desventajas. Para determinar la forma del invernadero se debe tener en cuenta varios factores, para elegir el tipo tenemos:

Gótico: Los invernaderos de estilo gótico tienen un techo arqueado que se asemeja a una hoja al revés. La forma del techo repele la lluvia y la nieve, lo que lo hace ideal si vive en un área con fuertes nevadas. Acristalamiento, doble poli, PE y vidrio son materiales ideales para este estilo.

Marco en A: El invernadero con marco en A es un diseño de invernadero estándar con una distribución uniforme en la parte inferior y dos lados inclinados de igual longitud para formar un techo en punta. Esta amplia base ocupa mucho espacio en el suelo. El flujo de aire y la maniobrabilidad son menores, pero es un diseño simple, se puede hacer con casi cualquier material y maneja bien la nieve pesada.

Marco de aro: Esta opción está muy extendida. Tiene una construcción de medio cilindro formada en un túnel largo. Es menos costoso de instalar y tiene un mayor control del flujo de aire interior que un marco en A. Sin embargo, las fuertes nevadas pueden causar daños estructurales.

A dos aguas: Este es un tipo común de invernadero. Tiene los lados rectos y un techo triangular; es lo suficientemente alto para moverse y recibe la mayor cantidad de luz solar. Debido al diseño diagonal, la nieve no tiende a acumularse. La estructura física es fácil de construir y está hecha de varios materiales, incluidos paneles de plástico rígido o vidrio.

1.4.2 Definición de servicios de computación en la nube

La manera más sencilla de explicar la computación en la nube es el acceso de programas y datos a través de internet en lugar de un servidor o recursos de computación propios. Con la computación en la nube los usuarios pueden acceder a archivos y usar aplicaciones desde cualquier dispositivo que pueda acceder a Internet. Una definición más formal es la proporcionada por NIST (National Institute of Standards and Technology) la cual indica que: La computación en la nube se define como un modelo para habilitar el acceso a la red de forma ubicua, bajo demanda y conveniente. Estos recursos (como redes, servidores,

almacenamiento, aplicaciones y servicios) son configurables, pueden ser proporcionados y liberados rápidamente con muy poco esfuerzo por parte del administrador o interacción con el proveedor de servicios. [8]

Figura 1.1 Varios servicios brindados por servicios en la nube.

1.4.3 Características de sistemas de computación en la nube

La computación en la nube se basa en recursos compartidos de cómputo en vez de servidores locales o dispositivos personales. Algunas de sus características más destacadas son:

- Acceso bajo demanda: La computación en la nube permite a los usuarios acceder a recursos de computación como almacenamiento, procesamiento y aplicaciones según sea necesario sin necesidad de invertir en infraestructura local.
- Recursos compartidos: Utiliza una red de servidores remotos para alojar administrar datos y aplicaciones, permitiendo así que varios usuarios accedan y compartan estos recursos.
- Escalabilidad: Permite a los usuarios escalar fácilmente el uso de recursos de cómputo según las necesidades del usuario sin necesidad de inversión en hardware o software adicionales.
- Flexibilidad: Permite a los usuarios agregar o eliminar fácilmente recursos de computación en función del cambio de necesidades sin necesidad de inversión inicial o compromiso a largo plazo.

- Pago por uso: Opera primordialmente en una base de pago por uso, donde los usuarios solo pagan por los recursos que consumen en lugar de tener que invertir en infraestructura costosa de antemano.
- Virtualización: La computación en la nube se basa en tecnología de virtualización que hace posible que varios usuarios compartan los mismos recursos físicos como servidores y almacenamiento sin interferir en los datos u operaciones del otro usuario.

1.4.4 Modelos de servicio para sistemas de computación en la nube

Existen tres tipos principales de modelos de servicio proporcionados al consumidor o cliente para sistemas basados en la nube, estos son: Infraestructura como Servicio (IaaS), Plataforma como Servicio (PaaS) y Software como Servicio (SaaS).

- Infraestructura como Servicio (IaaS). Brinda al consumidor la capacidad de procesamiento, almacenamiento, redes, además de otros recursos informáticos fundamentales donde el consumidor puede desplegar y ejecutar software arbitrario que puede incluir aplicaciones y sistemas operativos. El cliente no administra o controla la infraestructura interna de la nube, pero tiene control sobre sistemas operativos, almacenamiento, y aplicaciones desplegadas, además tiene la posibilidad limitada de seleccionar componentes de la red como firewalls. Permite a los usuarios rentar estos recursos según sea necesario, sin necesidad de invertir en su propia infraestructura.
- Plataforma como Servicio (PaaS). Facilita al usuario del servicio la capacidad de desplegar en la infraestructura de la nube aplicaciones creadas o adquiridas por el cliente, a través del uso de diferentes lenguajes de programación, librerías, servicios, y herramientas que sean soportadas por el proveedor. El consumidor no puede administrar o monitorear la infraestructura incluyendo: red, servidores, sistemas operativos o almacenamiento, pero tendrá control sobre la aplicación desplegada y las posibles opciones de configuración para la aplicación ambiente que hospeda el programa del cliente. Incluye la infraestructura subyacente, así como herramientas y servicios para construir y ejecutar aplicaciones.
- Software como Servicio (SaaS). Provee al consumidor con la capacidad de to usar las aplicaciones del proveedor que han sido desplegadas en la nube. Las aplicaciones son accesibles desde diferentes dispositivos del cliente a través de diferentes interfaces como pueden ser navegadores de internet o un programa. El

consumidor no tiene control sobre la infraestructura incluyendo la red, servidores, sistemas operativos, almacenamiento, o las capacidades de las aplicaciones con excepción de la configuración limitada de los servicios del usuario. SaaS permite a los usuarios utilizar estas aplicaciones a través de internet, sin la necesidad de instalarlas o mantenerlas en sus propios dispositivos.

Cada uno de estos modelos de servicio proporciona diferentes niveles de acceso y control sobre los recursos basados en la nube. IaaS proporciona el nivel más básico de acceso, mientras que PaaS y SaaS se basan en IaaS para proporcionar capacidades más avanzadas para construir y ejecutar aplicaciones en la nube.

1.4.5 Modelos de despliegue para sistemas de computación en la nube

Los modelos de despliegue se refieren a las formas en que se ofrecen los servicios basados en la nube a los usuarios. Hay tres principales modelos de despliegue para servicios en la nube: público, privado y híbrido.

- Público. La implementación en la nube pública hace referencia a servicios basados en la nube que se ofrecen al público en general a través de internet. Estos servicios suelen ser propietarios y operados por un proveedor externo, los usuarios pueden acceder a ellos en base al pago por uso.

- Privado. La implementación en la nube privada se refiere a servicios basados en la nube que operan exclusivamente para una sola organización. Estos servicios suelen estar ubicados en las propias instalaciones de la organización o en un centro de datos tercero dedicado, y no son accesibles al público en general.

- Híbrido. La implementación de la nube híbrida se refiere a una combinación de servicios en la nube pública y privada, donde algunos recursos son proporcionados por un proveedor de nube pública y otros son proporcionados por una nube privada. La nube híbrida permite a las organizaciones aprovechar los beneficios de ambos tipos de nubes; la escalabilidad y flexibilidad de la nube pública junto con el control y seguridad que brinda la nube privada.

Cada modelo de implementación tiene sus propias ventajas y desventajas, estas deben considerarse según las necesidades y requisitos específicos de la organización. La implementación de la nube pública suele ser la opción más rentable y escalable, pero es posible que no proporcione el mismo nivel de control y seguridad que la implementación de la nube privada. La implementación de la nube privada ofrece más control y seguridad, pero puede ser más costosa y menos flexible que la implementación de la nube pública. La

implementación de la nube híbrida permite a las organizaciones equilibrar los beneficios y los inconvenientes de la implementación de la nube pública y privada.

1.4.6 Arduino Yun y Yun Shield

Arduino Yun es una tarjeta de microcontrolador que combina un ATmega32u4 y un Atheros AR9331. La placa cuenta con un sistema operativo Linux basado en OpenWrt que recibe el nombre de Linino OS, y tiene características como conectividad Ethernet y Wi-Fi incorporadas, un puerto USB-A, una ranura para tarjeta micro-SD, 20 pines de entrada/salida digital (de los cuales 7 pueden ser usados como salidas PWM y 12 como entradas analógicas), un oscilador de cristal de 16 MHz, una conexión micro USB, un puerto ICSP y 3 botones de reinicio. [9]

Arduino Yun Shield amplía las capacidades de la placa Arduino con la potencia de un sistema basado en Linux que permite aplicaciones y conexiones de red avanzadas. El sistema Linux se basa en el procesador Qualcom Atheros AR9331 System on a Chip y OpenWrt-Yun, una distribución de Linux basada en OpenWrt. El Shield proporciona un puerto Ethernet con cable, una interfaz Wifi y un puerto USB-A. El Arduino Yun Shield es compatible con todas las placas Arduino con diseño R3: Arduino Uno, Arduino Leonardo, Arduino Mega 2560, Arduino Due, Arduino Zero. [10]

El modelo de la placa usada para este trabajo será la Dragino Yun Shield modelo v2.4. La placa cuenta con las siguientes características.

- Sistema operativo Linux de código abierto (OpenWrt)
- Bajo consumo de energía
- Es compatible con la versión de Arduino IDE 1.5.4 o posterior, permite al usuario programar, depurar o cargar proyectos a la placa Arduino mediante el IDE.
- Administrado a través de interfaz gráfica Web, SSH o mediante de LAN/Wifi
- El software se actualiza usando la red
- Cuenta con servidor web
- Permite la conexión a Internet mediante el puerto LAN, adaptador Wi-Fi o 3G.
- Tiene compatibilidad con memoria USB para almacenar proyectos en Arduino Yun.
- Su diseño resistente a fallas brinda un sistema robusto
- Compatibilidad con Arduino Leonardo, Uno, Duemilanove, Diecimila, Mega [11]

Para nuestro caso se usará un Arduino Uno con Yun Shield para enviar los valores medidos a un sistema en la nube. Arduino Yun tiene la habilidad de conectarse al internet y enviar datos a servidores remotos, además de otras características lo convierte en el candidato adecuado para este proyecto.

Figura 1.2 Placa Arduino Yun.

La diferencia principal entre el Arduino Yun y Arduino Yun Shield es que el primero tiene un microcontrolador integrado en donde corre el sistema operativo, mientras que para el Arduino Yun Shield el sistema operativo corre en el Shield y la energía será provista por Arduino Uno.

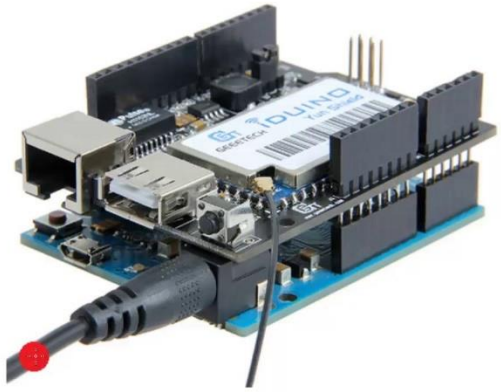


Figura 1.3 Arduino Yun Shield incorporado con Arduino Uno.

Ambos dispositivos tienen diferencias mínimas en términos de procesamiento. También presentan algunas características diferentes como incorporación de memoria microSD, estas no son de relevancia para el desarrollo del proyecto. El Arduino Yun Shield + Arduino Uno y el Arduino Yun tienen la misma funcionalidad en términos de conectividad a internet y posibilidad de programación avanzada que son de gran importancia para el proyecto. Otro motivo para usar la placa Arduino Yun Shield es debido a que la placa Arduino Yun al

momento de desarrollo del proyecto no se encontraba al mercado. Por estos motivos a pesar de que el proyecto se realizó usando Arduino Yun Shield + Arduino Uno se referirá a estos dispositivos como Arduino Yun ya que nuevamente, las diferencias son mínimas.

1.5 Proveedores de servicios en la nube

Entre los principales proveedores de servicios en la nube tenemos Amazon Web Services (AWS), Microsoft Azure y Google Cloud Platform. Otras opciones que cuentan con gran popularidad son IBM Cloud y Alibaba Cloud. Los proveedores que han sido nombrados principalmente proveen infraestructura como servicio. A continuación, se hablará los proveedores de servicios en la nube que han sido tomados en cuenta para el desarrollo del trabajo.

1.5.1 Amazon Web Services (AWS)

En 2006, Amazon Web Services (AWS) inició la oferta de servicios de infraestructura para tecnologías de la información a empresas bajo la forma de servicios en línea, hoy en día conocidos como computación en la nube. AWS actualmente brinda una plataforma de infraestructura en la nube altamente confiable, con escalabilidad y de bajo costo que impulsa a cientos de miles de empresas en 190 países alrededor del mundo. [12]

Figura 1.4 Logo de la plataforma Amazon Web Services.

AWS cuenta con muchos servicios en la nube que se pueden usar en diferentes combinaciones para ser adaptadas a las necesidades de empresas, organizaciones o individuos. Son usados para crear y ejecutar aplicaciones o aplicaciones web. Entre los principales servicios con los que cuenta AWS tenemos:

- Amazon Elastic Compute Cloud (EC2): Se refiere a un servicio de computación en la nube que provee recursos computacionales escalables y bajo demanda. Permite de manera rápida desplegar máquinas virtuales (también llamadas instancias) con gran variedad de sistemas operativos y configuraciones para ejecutar aplicaciones.

Permite escoger el tipo adecuado de instancia y su tamaño para la carga de trabajo necesaria, el pago se lo realiza para los recursos que han sido utilizados.

- Amazon Simple Storage Service (S3): Hace referencia a un servicio de almacenamiento escalable, durable y seguro. Puede guardar cualquier tipo de dato, esto incluye: texto, imágenes, videos y respaldos. Tiene gran popularidad para el alojamiento de activos estáticos que son usados para sitios web y el almacenamiento de datos usados por aplicaciones. Es altamente escalable, permite almacenamiento y recuperación de cualquier cantidad de información en cuando se lo desee. El pago es realizado por los recursos utilizados.

- Amazon Relational Database Service (RDS): Referido como RDS provee servicio de base de datos completamente administrado facilitando la configuración, el funcionamiento y el escalamiento de las bases de datos relacionales en la nube. RDS admite motores de bases de datos populares como son MySQL, PostgreSQL y SQL Server. Se encarga de tareas como realizar copias de seguridad, aplicar parches y supervisión. Esto permite que el cliente pueda concentrarse únicamente en crear su aplicación. RDS es escalable y de alta disponibilidad, se puede agregar o quitar capacidad fácilmente para satisfacer las necesidades de la aplicación.

- Amazon Simple Queue Service (SQS): SQS brinda servicio de mensajería que facilita separar y escalar microservicios, sistemas distribuidos y aplicaciones que no cuentan con servidor. Permite envío y recepción de mensajes entre diferentes elementos de la aplicación, gestiona automáticamente la disponibilidad y durabilidad de sus mensajes. SQS es escalable y totalmente administrado, permite una cantidad ilimitada de mensajes sin preocuparse por la capacidad, el rendimiento o disponibilidad.

1.5.2 Microsoft Azure

Es una plataforma de computación en la nube que cuenta con un portal en línea el cual permite acceso y administración de servicios además de recursos en la nube. Esta plataforma es proporcionada por Microsoft. Los servicios y recursos incluyen almacenamiento de datos o procesamiento de estos según sea el requerimiento del cliente. Para obtener acceso a los recursos y servicios es necesaria una conexión a Internet y entrar a la página de Azure Portal. Esta plataforma comenzó a brindar servicios en 2010, sus servicios son bajo demanda y gran cantidad de empresas de alto nivel usan sus servicios. [13]

Figura 1.5 Logo de la plataforma Microsoft Azure.

Azure cuenta con una gran variedad de servicios los cuales pueden ser divididos en las categorías de: computacionales, red y de almacenamiento. Cuenta con gran número de centros de datos alrededor del mundo. Estos servicios permiten crear, desplegar y administrar aplicaciones o cargas de trabajo en la nube. A continuación, se presenta una lista de los servicios más importantes:

- Máquinas Virtuales: Estos servicios permiten crear máquinas virtuales usando el sistema operativo Windows, Linux o cualquier otra configuración en poco tiempo.
- Servicios en la Nube: Brinda la posibilidad de crear aplicaciones escalables en la nube. Cuando la aplicación es desplegada los aspectos como provisionamiento, balanceo de carga y monitoreo serán a cargo de Azure.
- Azure Service Fabric: Simplifica de gran manera el proceso de desarrollo de microservicios, que se es una aplicación que contiene otras aplicaciones más pequeñas agrupadas.
- Azure CDN: Una CDN es una red de distribución de contenidos, y como su nombre lo indica su función es entregar contenido a los usuarios. El servicio de CDN usa una red de servidores ubicados de manera estratégica alrededor del planeta para que los usuarios puedan recibir los datos tan pronto como sea posible.
- Redes Virtuales: Una red virtual permite que cualquier servicio de Azure se pueda comunicar otro servicio de manera privada y segura.
- Azure DNS: Permite alojar nuestros dominios DNS o dominios de sistema en Azure.
- Almacenamiento de disco: Permite elegir entre HDD discos duros mecánicos o SSD discos duros de estado sólido como opciones de almacenamiento para nuestras máquinas virtuales.

- Almacenamiento de Blobs: Es un servicio optimizado para almacenar una enorme cantidad de datos no estructurados, entre los cuales se incluyen texto e inclusive datos binarios.
- Almacenamiento de archivos: Es un servicio administrado que posibilita acceder a archivos a través del protocolo SMB.

1.5.3 Heroku

Heroku es una plataforma que permite desplegar, ejecutar y gestionar aplicaciones desarrolladas en lenguajes como Ruby, Node.js, Java, Python, Scala, Go y PHP. Para esta plataforma una aplicación es una colección de código fuente escrito en uno de estos lenguajes, usando alguna estructura de desarrollo, y una descripción de dependencias que indican al sistema de compilación las dependencias necesarias para compilar y ejecutar la aplicación. [14]

Figura 1.6 Logo de la plataforma Heroku.

Es un servicio de plataforma como servicio (PaaS) que permite a los desarrolladores crear, ejecutar y escalar aplicaciones en la nube. Es una plataforma totalmente administrada, lo que significa que el desarrollador no tiene que preocuparse por la infraestructura, los servidores o los sistemas operativos: puede concentrarse en crear e implementar sus aplicaciones.

Para desplegar aplicaciones es necesario utilizar Git que es un poderoso sistema de control de versiones distribuido, usado por gran cantidad de desarrolladores para administrar y versionar el código fuente. Heroku utiliza Git como medio principal para la implementación de aplicaciones. Cuando se crea una aplicación en Heroku, asocia un Git remoto generalmente llamado Heroku con el repositorio local de Git para su aplicación. El despliegue consiste en mover la aplicación de un sistema local a Heroku.

Los pasos generales para realizar el despliegue en Heroku son los siguientes:

1. Registrarse en Heroku: Para desplegar una aplicación en esta plataforma es necesario tener una cuenta. Heroku cuenta con planes en diferentes precios de acuerdo con las necesidades del desarrollador.
2. Instalar la línea de comandos de Heroku: Conocida como Heroku CLI es una herramienta que se usa para administrar las aplicaciones desde línea de comandos. La instalación se realiza en la máquina local donde está la aplicación.
3. Crear una nueva aplicación en Heroku: Para desplegar la aplicación es necesario crear la aplicación y esto se lo puede realizar a través de la línea de comandos instalada o a través de la página web.
4. Configurar la aplicación: Es necesario configurar la aplicación para que pueda ser ejecutada en Heroku. Esto puede incluir crear variables de ambiente, añadir paquetes o modificar el código fuente para que se adapte a la plataforma de Heroku.
5. Desplegar el código de la aplicación: Una vez la aplicación ha sido configurada, se despliega la aplicación a través de la línea de comandos. Se puede usar diferentes métodos como usar Git, Docker, o usando una versión compilada.
6. Monitoreo y escalamiento: Una vez desplegada la aplicación se puede monitorear el desempeño y escalar usando la línea de comando o a través de la página web de Heroku.

Estos pasos pueden ser algo diferentes dependiendo de marcos de trabajo especiales o requerimientos que tenga la aplicación.

1.5.4 PythonAnywhere

Es una plataforma basada en la nube que permite a los usuarios ejecutar y alojar aplicaciones en la nube. Está diseñada especialmente para desarrolladores en Python, para que estos puedan crear y alojar aplicaciones web, realizar análisis de datos y otras aplicaciones basadas en Python todo en la nube.

Figura 1.7 Logo de la plataforma PythonAnywhere.

Esta plataforma provee bastantes características y servicios para desarrollar o alojar las aplicaciones. [15] Entre las cuales se incluyen:

1. Ambiente de desarrollo basado en web. Este ambiente de desarrollo basado en web permite a usuarios escribir, probar y depurar su código en Python desde cualquier dispositivo que cuente con una conexión a internet.
2. Plataforma de alojamiento: Esta plataforma permite a los usuarios desplegar y alojar sus aplicaciones en Python en la nube. Incluye gran cantidad de marcos de desarrollo web, como son Django y Flask, así como soporte para sockets web y tareas asíncronas.
3. Plataforma para base de datos: Provee varias opciones para bases de datos entre las que se incluye MySQL, PostgreSQL, y MongoDB, esto permite a los usuarios guardar y administrar datos en la nube.
4. Programador de tareas: Proporciona un programador de tareas que permite a usuarios programar secuencias de código en Python para que sean ejecutados ya se en un horario específico o a intervalos específicos.
5. Variedad de integraciones: Permite integrar herramientas y servicios populares como Git y Slack, esto permite a usuarios trabajar sin problemas con su flujo de trabajo existente.

2 METODOLOGÍA

El despliegue de una página web que permita monitorear las variables en un sistema de invernadero usando servicios en la nube considera múltiples pasos. Primero se debe considerar la creación de la página web la cual usará los lenguajes de programación JavaScript, CSS y HTML. El despliegue se lo realizará en primera instancia de manera local usando Flask el cual es un marco de desarrollo web basado en el lenguaje de programación Python. Luego se debe escoger servicio que provisione servicios en la nube, anteriormente fue observado que se debe considerar el modelo de servicio que sea adecuado para nuestro caso. El proveedor de servicio en la nube nos proveerá un dominio para la página web y permitirá luego de haber creado la cuenta subir los archivos necesarios a sus servidores para el despliegue. Una vez comprobado que la página web funcione de manera adecuada en la nube se realizará las conexiones necesarias hacia la base de datos, cabe mencionar que no es imprescindible que la página web y la base de datos estén alojados en un mismo proveedor de servicios en la nube. Finalmente se realizará la conexión del Arduino Yun con la página web haciendo usos de formularios web, estos nos permitirán enviar los datos hacia la página web, almacenarlos en la base de datos para ser posteriormente presentados al usuario que realice el monitoreo.

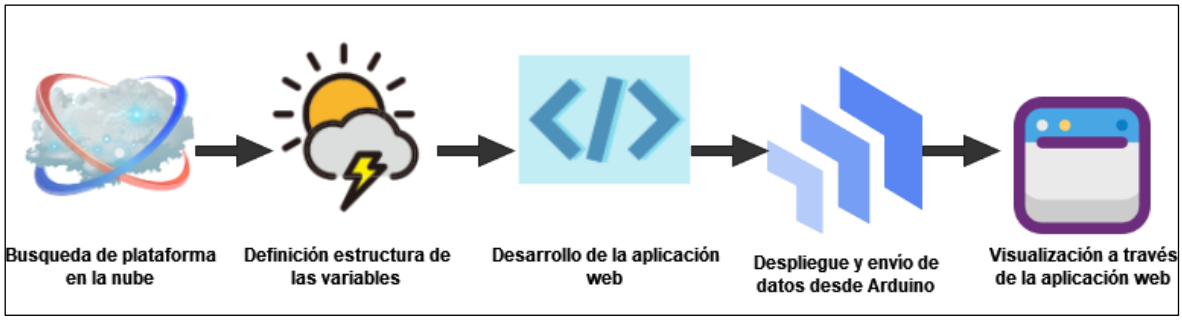


Figura 2.8 Etapas del desarrollo del proyecto.

Esta figura será la guía usada para identificar las distintas etapas de desarrollo en las cuales se encuentra este proyecto.

2.1 Consideraciones iniciales

Para el desarrollo del presente trabajo consideramos que las variables han sido tomadas, procesadas y están listas para ser transmitidas desde el Arduino hacia la base de datos en la nube. Las variables que se recolectarán son: temperatura, humedad, nivel de CO2, nivel del PH y luminosidad. Para verificar el funcionamiento apropiado se realizará simulación

de envió de los datos usando un script implementado en Arduino que tomará los valores promedio de las variables antes mencionadas y les agregará o restará cantidades aleatorias que se encuentren en el rango aceptable para las plantaciones. Para determinar estos valores se consultará la literatura correspondiente.

Se ha decidido por el marco de desarrollo de aplicaciones web Flask debido a que provee herramientas y características muy útiles para nuestro caso. Permite determinar rutas, manejar solicitudes y respuestas, además de integraciones con bases de datos lo permite crear aplicaciones web muy útiles, así como dinámicas.

2.2 Selección de la plataforma en la nube

Esta es la primera etapa del desarrollo del desarrollo del proyecto. Como parte de esta etapa se han definido pruebas de concepto en las cuales se maneja las distintas plataformas, aquí se consideran diferentes factores para su uso como son la facilidad de crear la cuenta, documentación y si requiere un método de pago. Aunque no se ha definido como parte de un objetivo que el desarrollo del proyecto sea totalmente libre de costo ha sido un enfoque tomado al elegir la plataforma. Se consideran niveles de servicio gratuito o estudiantiles sin costo como un factor a favor de las plataformas.

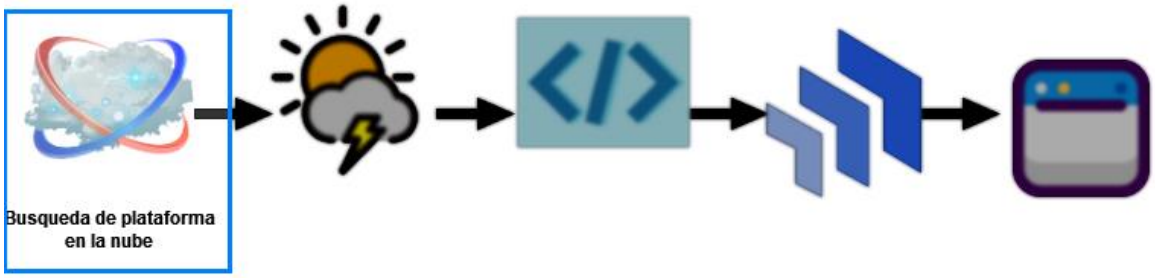


Figura 2.9 Primera etapa: Búsqueda de plataforma en la nube.

El proyecto consiste en el envío de datos por parte de la plataforma Arduino un servicio en la nube para que estos sean mostrados a través de una interfaz web. Para esto es necesario contar con un servicio que permita desplegar la aplicación, es decir provea alojamiento y acceso desde internet. Por otro lado, la aplicación debe contar con una base de datos que permita acceder a datos históricos e interactúe con la aplicación. Estos son los requisitos mínimos con los que debe contar el servicio en la nube.

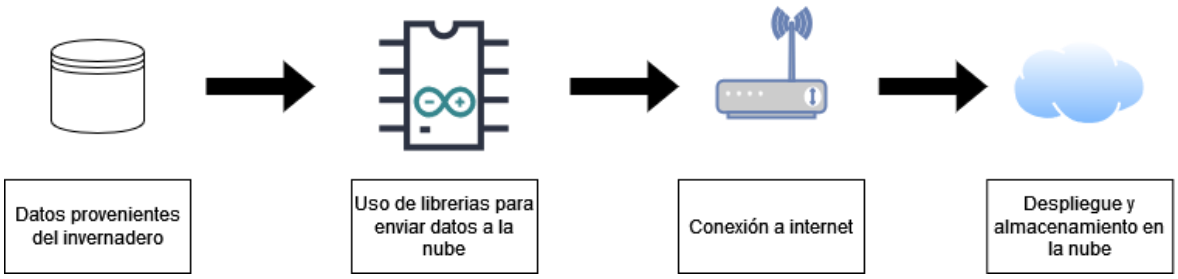


Figura 2.10 Esquema de funcionamiento del proyecto.

Es necesario para el proyecto contar al menos con una base de datos y alojamiento para la aplicación web. Esto se puede lograr mediante la creación de una máquina virtual, instalar las librerías y base de datos necesarios para desarrollar la aplicación web. Se ha visto que las plataformas AWS y Azure proveen máquinas virtuales, además existe la posibilidad de usar los servicios que brindan como alojamiento de la aplicación y la base de datos. Las plataformas Heroku y PythonAnywhere proveen servicios específicos en donde se aloja la aplicación web y la base de datos. A continuación, se presenta una lista comparativa entre ambos enfoques.

2.2.1 Despliegue mediante el uso de una máquina virtual

El despliegue de una aplicación web usando una máquina virtual en la nube es crear una máquina virtual en uno de los servicios en la nube, instalar y configurar en dicha máquina virtual la aplicación. Una máquina virtual es una solución en software que emula un computador físico que permite ejecutar un sistema operativo y aplicaciones en la nube. [16] A continuación se presenta una lista ventajas y desventajas del despliegue mediante el uso de una máquina virtual.

Tabla 2.1 Ventajas y desventajas de desplegar una aplicación web usando una máquina virtual en la nube

Ventajas	Desventajas
Flexibilidad: Debido a que se puede personalizar y configurar el sistema operativo, sus bibliotecas y dependencias según sea necesario. Esto provee máxima flexibilidad para ejecutar la aplicación.	Complejidad: Requiere instalar más elementos y mayor configuración que el uso de servicios en la nube específicos. Dicha configuración puede llevar mucho tiempo y ser complejo
Control: se tiene control total sobre el sistema operativo, esto permite administrar la aplicación y las dependencias según sea necesario.	Administración: La máquina virtual requiere administración y mantenimiento continuos. Esto involucra instalación de parches y actualizaciones del sistema operativo, así

	como de sus dependencias. Esto puede ser una carga para desarrolladores
Compatibilidad: La máquina virtual permite ejecutar la aplicación en el mismo sistema operativo en el cual se ha desarrollado la aplicación de manera local que puede tener requerimientos específicos.	Costo: Suele ser mas costosa de ejecutar que usar servicios en la nube mas específicos, especialmente si es necesario posteriormente escalar la aplicación de mayor manera o menor manera con frecuencia.

2.2.2 Despliegue mediante el uso de servicios específicos

En este caso se usan servicios en la nube ya construidos para alojar y ejecutar la aplicación, en lugar de crear y administrar máquinas virtuales. Esta opción es más simple y con mejor costo para desplegar la aplicación. No es necesario preocuparse acerca de la configuración y administración de la infraestructura o sistema operativo en la cual la aplicación corra.

Tabla 2.2 Ventajas y desventajas de desplegar una aplicación web usando servicios específicos en la nube

Ventajas	Desventajas
Simplicidad: Al usar servicios específicos en la nube ya sea una base de datos administrada o plataformas de computación sin servidores suelen ser mas sencillos de usar que los desplegados en una máquina virtual, debido a que no se debe configurar o administrar la infraestructura.	Flexibilidad limitada: Usar un servicio específico en la nube puede ser menos flexible que usar una máquina virtual ya que nos debemos limitar a los servicios o configuraciones que disponga el proveedor de servicios en la nube.
Rentabilidad: El uso de servicios específicos suele ser más rentable que el uso de una máquina virtual ya que solamente necesita un conjunto limitado de servicios.	Dependencia del proveedor: Al ser servicios específicos provistos por un proveedor, se depende del proveedor para mantener y actualizar los servicios, esto presenta un riesgo si existen interrupciones.
Seguridad: Los proveedores de servicios en la nube suelen provee una amplia gama	Bloqueo potencial en proveedor: Al usar un servicio específico puede ser que nuestra aplicación se vuelva dependiente de dichos

de funciones y controles para mantener a la aplicación y sus datos seguros	servicios y si se desea migrar de servidor se debe reescribir la aplicación.
Escalabilidad: Esta clase de servicios suelen estar diseñados para ser altamente escalables, esto permite que dependiendo de la carga la aplicación sea escalada de mayor manera o de menor manera dependiendo de la demanda.	Costo potencial: Si se cuenta con gran cantidad de servicios diferentes puede ser mas rentable usar máquinas virtuales y dejar que esta procese dichos servicios.

Para poder determinar qué plataforma es apropiada para la realización del proyecto preliminarmente se ha analizado y probado las plataformas mencionadas anteriormente. El uso de cualquiera de las plataformas necesita de una cuenta de usuario y en algunos casos definir un medio de pago. A continuación, se presentarán los pasos para crear una cuenta en los distintos servicios.

2.2.3 Creación de cuentas de usuario y revisión de características de las plataformas

A continuación, se presentarán los servicios que han sido considerados para la realización de este trabajo, se escogió dos plataformas que funcionan a manera de Infraestructura como servicio (IaaS) y Plataforma como servicio (PaaS).

2.2.3.1 Cuenta institucional y sus beneficios

La Escuela Politécnica Nacional brinda beneficios, entre ellos se encuentra un correo institucional. Dicha cuenta brinda 50GB de almacenamiento y 1TB disponible en un repositorio virtual de OneDrive para almacenar carpetas y archivos. Además, brinda acceso a herramientas colaborativas y manejo de grupos de distribución. [17]

Dentro de la página del centro de descargas y Software Académico se encuentran varias herramientas a las que se puede acceder de manera gratuita para fines académicos. Entre las ofertas se encuentra Microsoft Azure Dev Tools for Teaching, esta plataforma ofrece recursos académicos y la capacidad de crear proyectos o probar los servicios que ofrece Azure.

Muchas instituciones tienen acuerdos y contratos con organizaciones que proveen acceso especial o descuento para estudiantes. Estos acuerdos pueden incluir acceso a documentos académicos, bases de datos o descuentos en productos de software y

tecnología. Todos estos beneficios pueden ser accedidos a través de la dirección de correo electrónico que contiene “.edu” en su dominio.

Uno de los beneficios más relevantes para el desarrollo de este proyecto fue el brindado por GitHub dentro de su programa GitHub Student Developer Pack. Provee a los estudiantes de amplia variedad de herramientas, que permiten desplegar software y recursos. Este programa ha sido enfocado en el aprendizaje de los estudiantes para que desarrollen sus habilidades en desarrollo de software.

El paquete incluye cuenta de GitHub pro que cuenta con beneficios adicionales a las cuentas normales de GitHub. También cuenta con ofertas de muchos socios que brindan sus servicios con descuento u ofrecen niveles estudiantiles que incluyen uso sin costo durante un tiempo. Entre las principales empresas que cuentan ofertas tenemos:

- DigitalOcean: Plataforma de alojamiento en la nube que permite administrar, crear y escalar servidores virtuales. Ofrece 200$ en créditos para la plataforma por un año.
- Microsoft Azure: Acceso a servicios que brinda en la nube y recursos para el aprendizaje, no requiere tarjeta de crédito para su registro. Ofrece acceso a más de 25 servicios gratuitos en la nube y 100$ de crédito Azure.
- Heroku: Una herramienta flexible y fácil de usar que permite desplegar, correr y administrar aplicaciones. Brinda un crédito de 13$ mensuales durante 12 meses.
- GitHub Pages: Permite crear sitios web para proyectos. Permite el alojamiento directamente desde el repositorio de GitHub. Brinda un sitio por cuenta y sitios de proyecto ilimitados.

Además, se pueden nombrar varios sitios populares como: Namecheap, Name.com, JetBrains, Educative, FrontendMasters, Twilio, StreamYard. Que son pocas de las muchas ofertas que se encuentran dentro de este paquete y aunque no son de relevancia para este proyecto se recomienda explorar las distintas aplicaciones ofrecidas en este paquete.

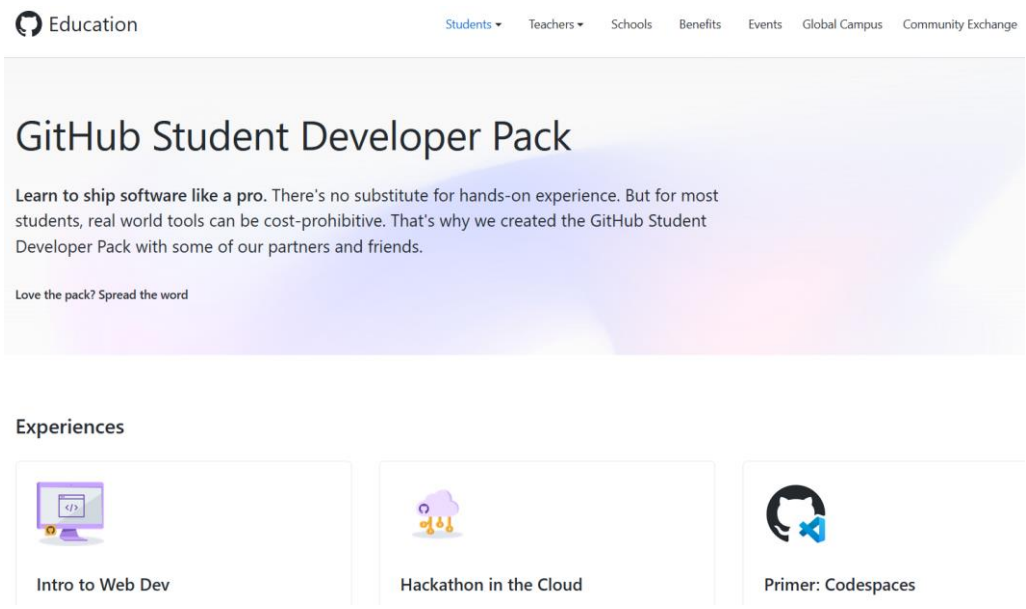


Figura 2.11 Página de ofertas disponibles en GitHub Student Developer Pack

2.2.3.2 Amazon Web Services

La primera plataforma probada fue Amazon Web Services a continuación se presentan los pasos seguidos para la creación de la cuenta.

1. Ingrese al portal de AWS a través de la dirección https://aws.amazon.com/ y de clic en “Crear una cuenta AWS”.
2. Ingrese la dirección de correo y un nombre de usuario y diríjase a continuar.
3. Un correo será enviado a la dirección de correo electrónico que se ingresó para el registro, el correo recibido tendrá un código que será ingresado para continuar con el registro.
4. Se requerirán datos personales, los llenamos y continuamos.
5. Solicitará datos de pago, será necesaria una tarjeta de crédito para el registro.
6. Finalmente se creará la cuenta de usuario.

Figura 2.12 Creación de una cuenta de usuario en AWS.

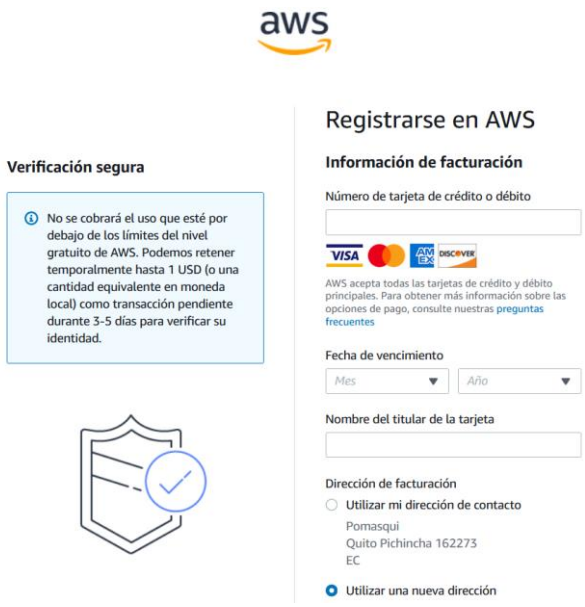


Figura 2.13 Registro de pagos en AWS.

Debido a que es necesario contar con una tarjeta de crédito para la creación de una cuenta o usar los servicios, esta plataforma fue descartada para el desarrollo del proyecto y se continuó explorando las demás opciones.

2.2.3.3 Microsoft Azure y Azure para estudiantes

El proceso de instalación es muy similar al de AWS. Para crear una cuenta en Microsoft Azure debemos seguir los siguientes pasos

1. Entrar la página principal de Microsoft Azure: https://azure.microsoft.com/en-us/ y damos clic a “Cuenta Gratuita”.
2. Allí tendremos dos opciones principales, debemos dar clic en: “Empiece gratis”.
3. Se mostrará una ventana que nos pedirá que registremos, debemos usar una cuenta en Outlook para realizar este paso.
4. Debemos ingresar información personal y dar a “Crear cuenta gratis”

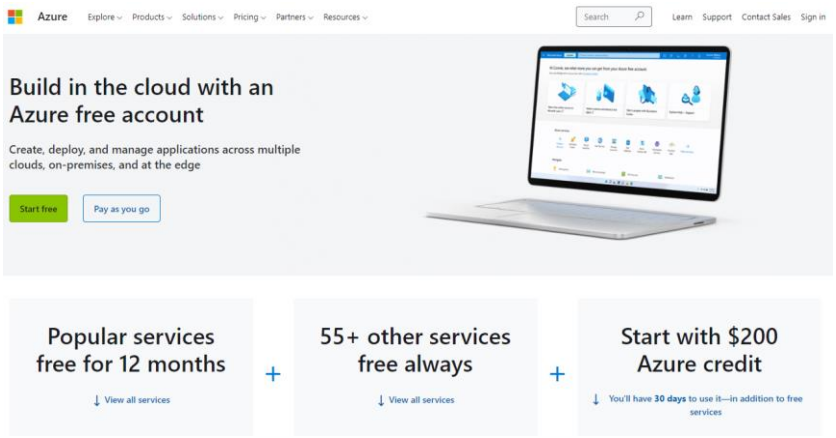


Figura 2.14 Portal de la plataforma Microsoft Azure.

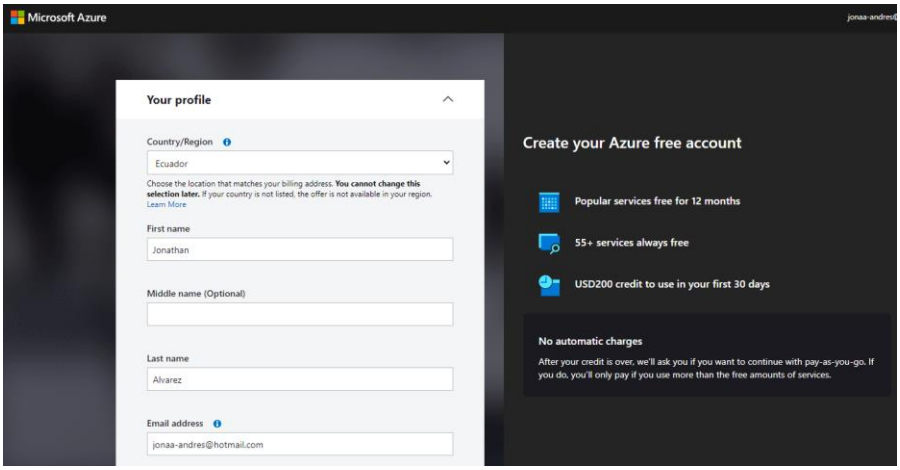


Figura 2.15 Ingreso de datos personales para la creación de la cuenta en Microsoft Azure.

Una vez creada la cuenta tendremos acceso al portal de Microsoft Azure donde se encontrará la siguiente ventana de bienvenida

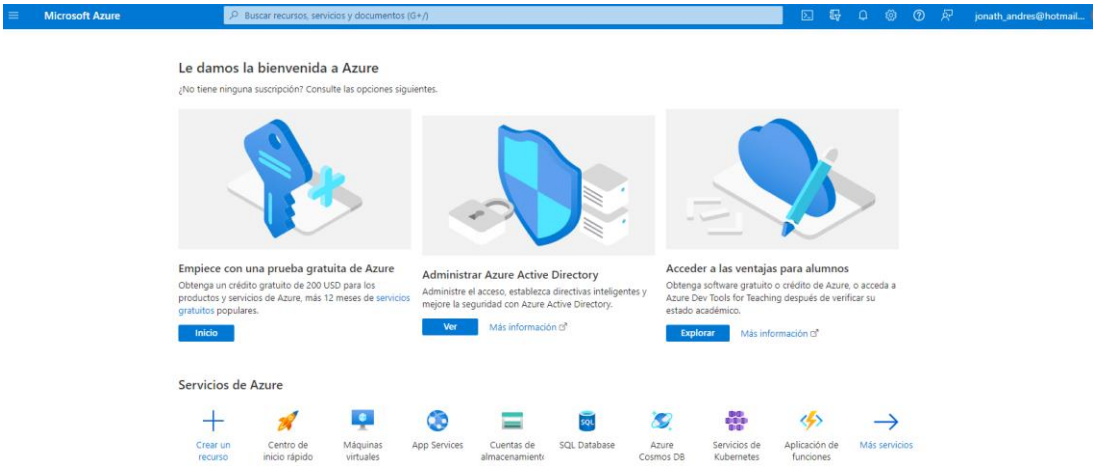


Figura 2.16 Pantalla de bienvenida de Azure Portal.

Se puede observar en la imagen que la plataforma ofrece tres opciones principales, una es una prueba gratis de Azure, la segunda se relaciona con la administración de Azure Active Directory que es un servicio que brinda Azure, finalmente tenemos la opción Acceder a las ventajas para alumnos. Al ingresar a esta opción nos encontramos con que existe una suscripción en la cual nos ofrecen 100 USD en créditos gratuitos que nos sirven para crear servicios de Azure y estarán disponibles por 12 meses.

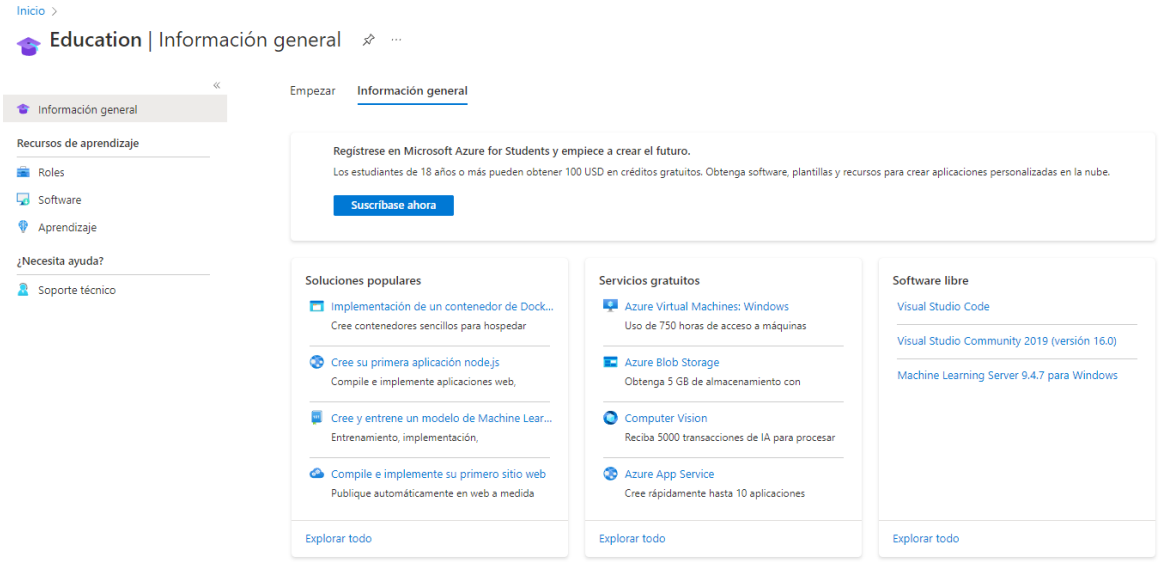


Figura 2.17 Suscripción para estudiantes.

Microsoft Azure para estudiantes es un programa de subscripción gratuita diseñada para estudiantes que se encuentren interesados en aprender acerca de computación en la nube y desarrollo de aplicaciones basadas en la nube. Esta subscripción permite a acceder a una gran variedad de servicios y herramientas que provee Azure; esto incluye máquinas virtuales, aplicaciones web y sistemas de bases de datos sin costo.

Para ser elegible para la suscripción Azure para estudiantes se debe contar con una dirección de correo electrónico provista por una institución educativa o haberse graduado en los últimos 12 meses. También será necesario seguir un sistema de verificación. [18]

Figura 2.18 Ingreso a Azure para Estudiantes.

Una vez realizada la verificación al seguir los pasos indicados y haciendo uso de nuestro correo institucional llegaremos nuevamente a la ventana de educación con nuestros créditos y el tiempo que tenemos antes de que caduque la oferta.

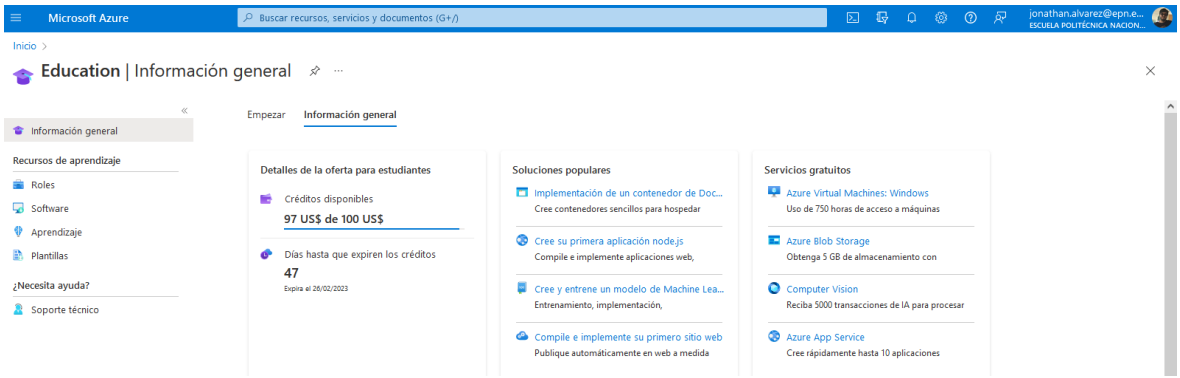


Figura 2.19 Ventana de Educación una vez verificada la cuenta institucional.

Como estudiante con acceso a la suscripción Azure para Estudiantes podremos:

- Aprender acerca de la computación en la nube y desarrollo de aplicaciones basadas en la nube.

- Crear proyectos y desarrollar aplicaciones web (como es el nuestro caso), aplicaciones móviles y otros proyectos.
- Configurar y usar máquinas virtuales para desarrollo, así como pruebas.
- Almacenamiento de datos mediante las múltiples opciones de bases de datos y servicios de análisis de datos.
- Colaborar en proyectos con compañeros y colegas mediante herramientas de productividad basadas en la nube. [19]

Si nos dirigimos hacia la opción de servicios gratuitos accederemos a servicios que son siempre gratis pero que cuentan con limitaciones ya sea en procesamiento, memoria, transacciones computacionales, etc. De igual manera estos servicios gratuitos son una buena manera de probar Azure, así como sus varias características y capacidades. Algunos de los servicios más importantes que contamos con la cuenta gratuita son:

- Máquinas virtuales: Creadas en Windows o Linux por 750 horas, cuenta con 5 GB de almacenamiento.
- Azure Dev Tools para Educadores: Es un programa que provee acceso a un amplio rango de servicios y herramientas para profesores y estudiantes que se enfocan en la enseñanza
- Azure Funciones: Este servicio permite ejecutar pequeños fragmentos de códigos que se denominan funciones. La nube los ejecuta sin tener que preocuparse por la infraestructura necesaria para la ejecución sirve para automatizar tareas.
- Azure IoT Hub: Este servicio permite conectarse de manera segura, monitorear y administrar dispositivos IoT a escala. Esto quiere decir que se puede usar este servicio para crear soluciones IoT de gran escala ya que permite el procesamiento de gran cantidad de datos provenientes de los dispositivos conectados.
- Azure App Service: Permite crear y alojar servicios web, móviles y API desde la nube. Permite escalar aplicaciones fácil y rápidamente. Se paga solamente por los recursos consumidos.

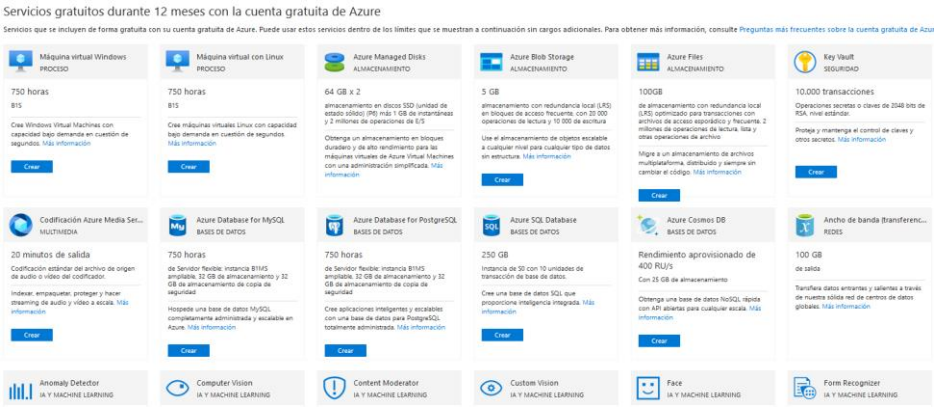


Figura 2.20 Algunos de los servicios gratuitos provistos por Microsoft Azure.

2.2.3.4 Heroku

Para crear una cuenta en Heroku se debe seguir los siguientes pasos:

1. Ir a la página de bienvenida de Heroku en https://www.heroku.com/ y dar clic en la opción “Sign up”.
2. Se solicitarán los datos para la cuenta como son nombres, dirección de correo, país e idioma entre otros.
3. Se enviará un correo electrónico al buzón que contendrá el enlace que se debe dar clic para verificar la cuenta.
4. Una vez verificado nuestro correo electrónico se puede ingresar a la cuenta usando dicho correo y contraseña

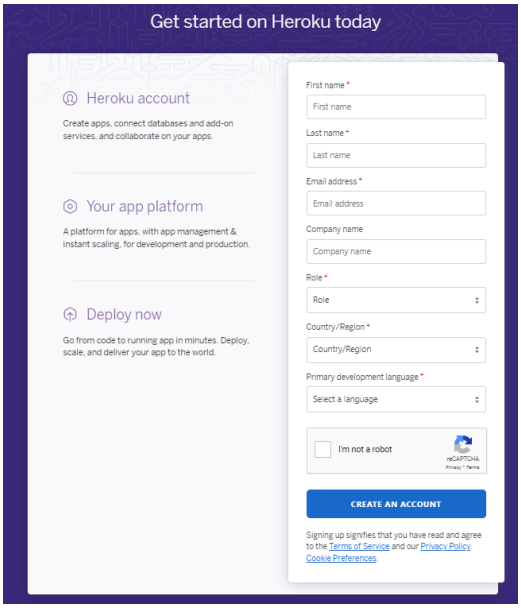


Figura 2.21 Datos necesarios para la creación de la cuenta en Heroku.

Heroku en las primeras etapas del desarrollo del proyecto ofrecía un nivel de servicio gratuito que permitía implementar y ejecutar aplicaciones de manera gratuita. Aunque estos recursos eran limitados contaban con las características necesarias para el despliegue de una simple aplicación de una manera muy fácil.

La última versión de Heroku no contiene un nivel gratuito y requiere del registro con tarjeta de crédito. Heroku de igual manera cuenta con una oferta para estudiantes que ofrece un crédito de 13 USD que estarán disponibles durante un año, para este registro es necesario ingresar un método de pago con una tarjeta de crédito.

Gracias a GitHub Student Developer Pack es posible acceder al bono ofrecido por Heroku a estudiantes, luego del registro es necesario dirigirse a la siguiente dirección web: https://www.heroku.com/github-students/. Al ingresar a “Get the Student offer” luego de la verificación de la cuenta institucional por parte de Heroku, verificación de un método de pago y aceptación de la aplicación.

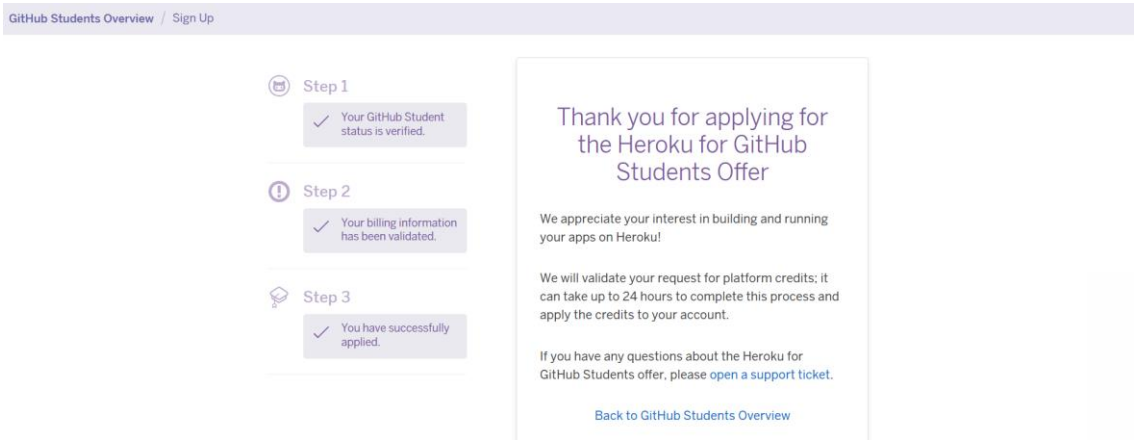


*Figura **Error! Use the Home tab to apply 0 to the text that you want to appear here.**.22 Proceso para acceder a la oferta de estudiantes de Heroku.*

2.2.3.5 PythonAnywhere

Para el registro en PythonAnywhere seguimos los siguientes pasos:

1. Entrar a la página web https://www.pythonanywhere.com/ y dar clic en el botón “Pricing and signup”.
2. Encontraremos múltiples opciones de registro, usaremos el nivel de servicio Begginer el cual es gratuito.
3. Ingreso de la información solicitada que es nombre para el usuario, correo electrónico y contraseña.

4. Verificación de correo electrónico a través del enlace enviado al correo provisto en pasos anteriores.

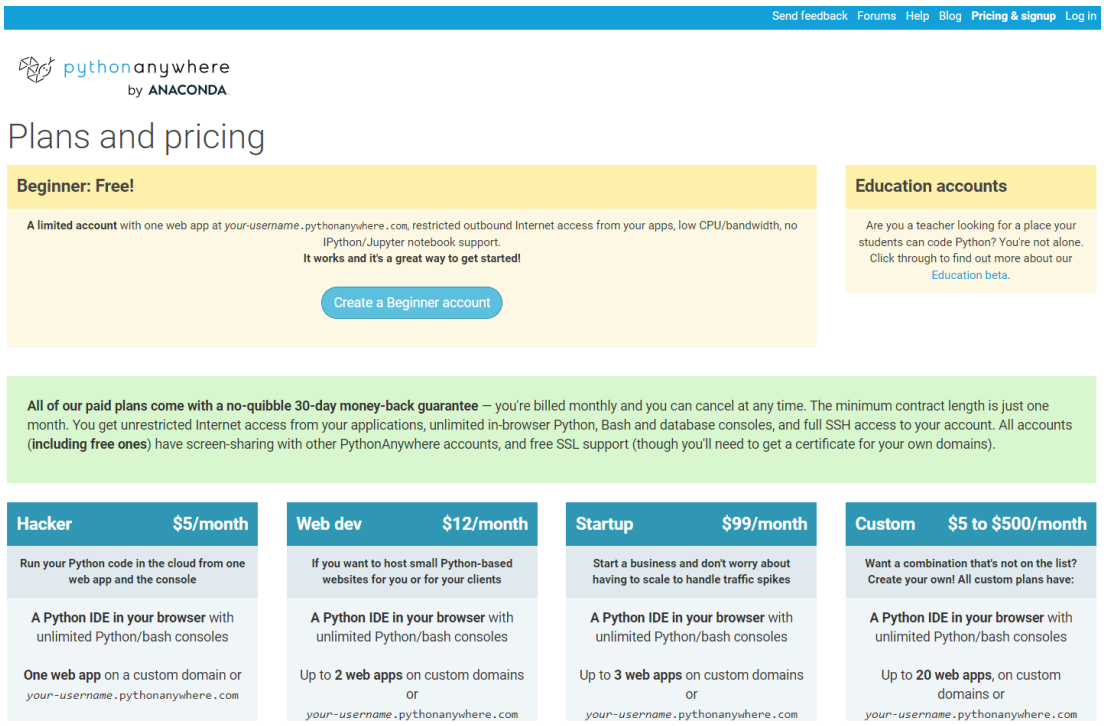


Figura 2.23 Planes y precio de la plataforma PythonAnywhere.

PythonAnywhere ofrece muchas características gratuitas para cuentas Beginner, nos permite a través de una interfaz web crear y ejecutar programas usando Python 3.7 y Python 3.8. Contaremos con un editor web que permite modificar scripts en Python además de archivos HTML, CSS y JavaScript.

Esta plataforma se centra en desarrollo basado en Python, aunque soporta otros lenguajes como Bash y MySQL. Esta plataforma se exploró con el despliegue de una de las primeras iteraciones de la aplicación, se presentaron problemas al realizar la conexión entre este servicio y Azure. Aunque parecía prometedor en primera instancia se optó por otras alternativas.

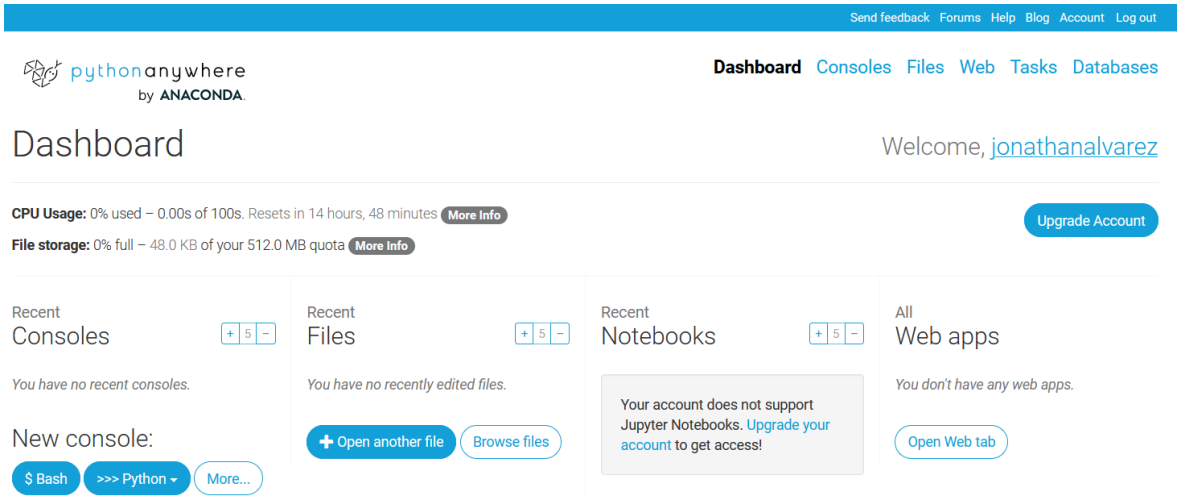


Figura 2.24 Página principal al ingresar a la cuenta de usuario.

2.3 Definición de variables y su estructura de envío

La siguiente etapa considera la interacción que tendrá el Arduino Yun con la aplicación localizada en la nube. Como se vio anteriormente la placa tiene Wifi incorporado lo que le brinda conexión a internet, además sus capacidades de programación avanzadas nos permitirán a horas programadas o cuando se reciban nuevas lecturas enviar los datos del sistema de invernaderos hacia el servicio en la nube.

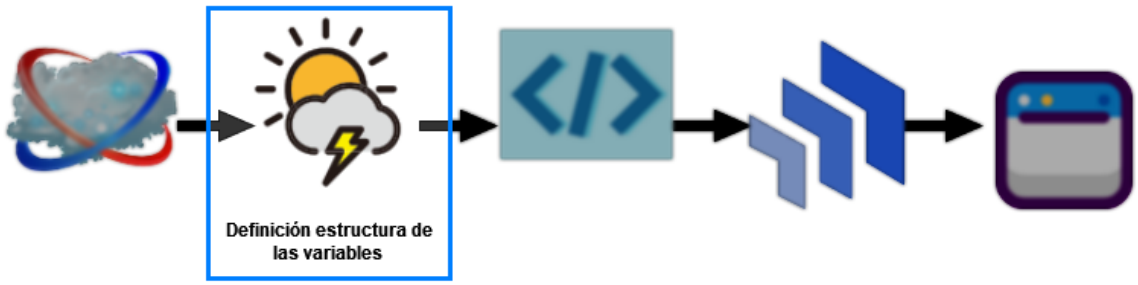


Figura 2.25 Segunda etapa: Definición de la estructura de las variables para su envío al servicio en la nube.

Este componente estudia el formato de las variables y el tipo de los valores. Cuando se realice el despliegue de la aplicación web en la nube y el almacenamiento de las variables en una base de datos que de igual manera estará en la nube, se asume que los datos que serán enviados han sido tratados y validados apropiadamente con anterioridad. Es decir, los valores enviados idealmente serán resultado de la captura realizada por otro componente.

2.3.1 Variables transmitidas

Las variables consideradas para ser enviadas a través de internet son: temperatura ambiental, humedad del ambiente, humedad del suelo, nivel de PH del agua, luminosidad.

Aunque no es de importancia el tipo de variables en si para este componente del proyecto de sistema de invernaderos, es importante que las variables sean transmitidas y almacenadas en un servidor en la nube. Se han elegido las variables mencionadas por que estos factores afectan de gran manera el crecimiento y desarrollo de las plantas. Además, se toma en cuenta valores y variables referentes a plantas medicinales para ilustrar el funcionamiento del proyecto. A continuación, se muestra una breve explicación de la importancia de cada una de las variables:

- Temperatura ambiental: Las plantas medicinales tienen un rango de temperatura en el cual pueden desarrollarse apropiadamente. Si la temperatura es muy alta o baja, es posible que no se realice la fotosíntesis que afecta el crecimiento y reduce la calidad de la planta.
- Humedad del ambiente: Las plantas requieren niveles específicos de humedad para prosperar. Alta humedad puede causar el crecimiento de hongos y moho, por otro lado, baja humedad puede inhibir el crecimiento.
- Humedad del suelo: El nivel de humedad de suelo apropiado es necesario para que las plantas crezcan adecuadamente. Riego excesivo o insuficiente puede provocar que la raíz se pudra o se marchite, esto causa que el crecimiento no sea adecuado y reduce la calidad de la planta. [20]
- Nivel de PH: Es importante considerar este factor debido a que indica la disponibilidad de nutrientes esenciales para la planta. El nivel de PH es la medida de la acidez o alcalinidad del suelo. [21]
- Luminosidad: Las plantas requieren un nivel de luz apropiado para crecer. Algunas plantas requieren sol otros tipos necesitan sombra parcial. Este factor influye en la capacidad de desarrollarse adecuadamente de la planta.

Los siguientes valores son cantidades promedio para las variables mencionadas y sus variaciones. Estos valores se han encontrado en distintas referencias. Al no ser parte de este componente del sistema de invernaderos ya que se asume valores tratados no se ha dado mayor rigurosidad a dichos valores, sino que se los tiene como referencia general para demostrar el funcionamiento de este trabajo.

Tabla 2.3 Valores de las variables usadas como valores para el envió de datos. [20] [21] [22]

Variable	**Rango**
Temperatura ambiental	28 – 41°C
Humedad del ambiente	54 – 89 %
Humedad de suelo	40 – 60 %
Nivel de PH	6.0 – 7.5
Luminosidad	1.05 – 100 %

2.3.2 Método de transmisión y estructura de envío

Como ya se ha mencionado anteriormente Arduino Yun cuenta con conexión a internet y funciona con un sistema operativo basado en Linux. Dicho sistema operativo le permite al dispositivo hacer solicitudes a través de la herramienta de línea de comandos llamada CURL.

2.3.2.1 CURL

Conocido como Cliente para URLs es una herramienta de línea de comandos usada en el envío datos hacia y desde servidores, soporta gran cantidad de protocolos entre los que encontramos: HTTP, FTP, SCP, SMTP, Telnet, etc. Se usa para descargar archivos, subir archivos a servidores y enviar o recibir datos desde APIs.

CURL se encuentra disponible en la mayoría de los sistemas operativos. Es una herramienta usada mediante línea de comandos, a través de una terminal. Puede ser usada en scripts lo que permite automatizar las tareas, esto la convierte en una herramienta muy usada en desarrollo web, administración de sistemas y análisis de datos. [23]

Para enviar los datos se usará el siguiente comando:

```
curl -k -X POST -H Content-Type: application/x-www-form-urlencoded -d
dataString url
```

Script 2.1 Uso de CURL para enviar datos hacia la aplicación en la nube.

La explicación de los parámetros usados es la siguiente:

- -k: Esta opción le indica a CURL que ignore los errores relacionados a certificados SSL. Le permite a CURL conectarse al servidor aunque el certificado sea invalido o haya expirado
- -X POST: Esta opción especifica que la solicitud será del tipo POST. Este método es usado para enviar datos hacia el servidor. [24]

- H Content-Type: application/x-www-form-urlencoded: Define el tipo de contenido e indica que los datos que serán enviados en la solicitud tendrán un formato de tipo “variable=valor”, y que cada valor será separado por ‘&’.
- -d dataString: Contendrá el cuerpo de la solicitud, al enviar los datos se reemplazará por valores de las variables definidas anteriormente.
- url: Indica la URL del servidor en donde se encuentra desplegada la aplicación

2.3.2.2 Estructura del envío

Una vez definido el método del envío es necesario seguir el formato definido por CURL que nos permitirá que los datos sean recibidos por la aplicación web para que sean almacenados en la base de datos. El comando especifica que deben tener la forma de ‘variable1=valor1&variable2=valor2’. Como parte de las pruebas del proyecto se definió la siguiente función que por un lado genera los datos teniendo en cuenta los valores de las variables del sistema de invernaderos investigados, y por otro construye los pares de valores para que sean incluidos en la solicitud.

```
// Esta función genera los valores teniendo en cuenta los rangos investigados
// Luego les da la forma ‘variable=valor’ seguido de & para completar el cuerpo de la solicitud
// Almacenamiento de datos se lo realiza usando variable global
void updateData() {
  dataString = “content=EnviadoArduino&”;
  dataString += “temperatura=”;
  dataString += String(random(14) + 28);
  dataString += “&”;
  dataString += “humedad=”;
  dataString += String(random(36) + 54);
  dataString += “&”;
  dataString += “humsuelo=”;
  dataString += String(random(21) + 40);
  dataString += “&”;
  dataString += “nivelPH=”;
  dataString += String(random(3) + 6);
  dataString += “&”;
  dataString += “luminosidad=”;
```

```
  dataString += String(random(100) + 1);
  Console.print(dataString);
}
```

Script 2.2 Generación y formato de los datos a ser enviados.

Esta función se encuentra implementada en el código que permite comunicar el Arduino y el servicio en la nube a través de internet.

2.4 Marcos de desarrollo para aplicaciones web y desarrollo local de la aplicación

Luego de haber solventado la necesidad del servidor web que nos permitirá alojar nuestra aplicación, resolver solicitudes y respuestas que realizaran los usuarios a nuestra aplicación web se debe escoger el marco de desarrollo en el cual se creara la aplicación web.

Los marcos de desarrollo de software son una colección de librerías y módulos que facilitan la creación y mantenimiento de una aplicación web. Cuentan con funcionalidad y estructura predefinidas, permitiendo a quien desarrolle enfocarse en escribir el código de la aplicación.

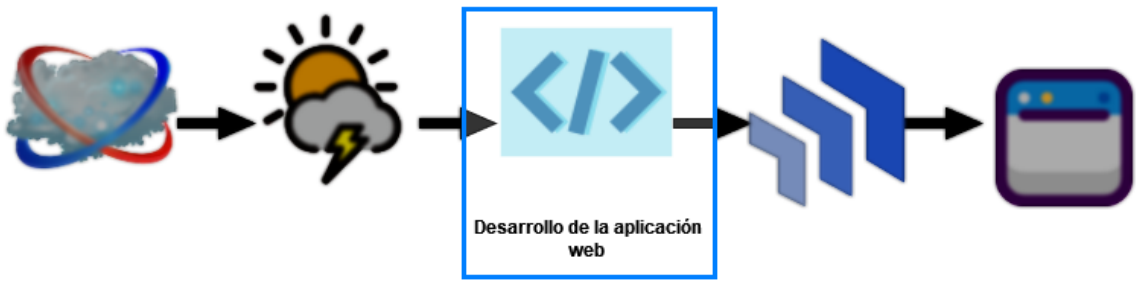


Figura 2.26 Etapa 3: Desarrollo de la aplicación web.

2.4.1 Justificación del lenguaje de programación y marco de desarrollo de la aplicación web

Como se mencionó anteriormente el autor de este trabajo cuenta con experiencia en el lenguaje de programación Python. A través de la carrera ha sido fomentado como un lenguaje potente y versátil para la solución de muchos tipos de problemas. En la investigación previa al desarrollo de este trabajo se encontró que este lenguaje de

programación cuenta con muchos marcos de desarrollo web que facilitarían el desarrollo de la aplicación.

2.4.1.1 Lenguaje de programación Python

Python es un lenguaje de programación de propósito general de alto nivel que es usado ampliamente para el desarrollo web, computación científica, análisis de datos, inteligencia artificial y más. Es conocido por su simplicidad, legibilidad y flexibilidad. Tiene una comunidad grande y activa, además cuenta con una gran cantidad de bibliotecas y marcos de desarrollo disponibles para una amplia gama de tareas. Python admite estilos de programación funcional, imperativo y orientado a objetos, además tiene un sistema de tipo dinámico y administración automática de memoria. [25]

A continuación, se listará algunos de los usos que se le puede dar a Python:

- Creación de scripts: Una de las formas más comunes de usar Python es escribir scripts que automaticen tareas repetitivas o realicen otros tipos de procesamiento de datos. Se pueden ejecutar desde la línea de comandos y pueden ser usados para automatizar tareas como el procesamiento de datos y mucho más.
- Shell interactivo: Python también viene con un Shell interactivo, que permite escribir y probar código de Python en tiempo real. Esto puede ser útil para experimentar con código nuevo o solucionar problemas de código existente.
- Jupyter Notebooks: Son un entorno de desarrollo interactivo basado en la web que permite escribir y ejecutar código Python, así como agregar texto, imágenes y otros tipos de medios. Opción popular enfocada en la ciencia de datos y la computación científica, permite compartir fácilmente trabajo y colaborar con otros.
- Creación de aplicaciones independientes: Se puede usar para crear aplicaciones independientes ya sea aplicaciones de escritorio o juegos. Se pueden usar marcos y bibliotecas populares como PyQt y Pygame para crear aplicaciones de escritorio y juegos, respectivamente.
- Desarrollo web: Python es una opción popular para el desarrollo web y hay muchos marcos de desarrollo web disponibles para Python.
- Desarrollo de modelos de aprendizaje automático: Python se usa ampliamente en el campo del aprendizaje automático, con bibliotecas populares como TensorFlow, PyTorch y scikit-learn. Se puede usar esas bibliotecas para crear, entrenar y probar modelos de aprendizaje automático. [26]

2.4.1.2 Marcos de desarrollo en Python

Python es una opción popular para el desarrollo web debido a su simplicidad, flexibilidad y a una gran comunidad activa. Hay muchos marcos de desarrollo web disponibles para Python, como Django, Flask, Pyramid y Tornado, cada uno con sus propias fortalezas y debilidades.

Algunos marcos de desarrollo web populares para Python incluyen:

- Django: Es un marco de desarrollo web de alto nivel que fomenta el desarrollo rápido con un diseño limpio y funcional. Sigue el patrón arquitectónico Modelo-Vista-Controlador (MVC) y proporciona un ORM, una interfaz de administración integrada y muchas otras características útiles.
- Flask: Es fácil de aprender y usar. No incluye un ORM ni una interfaz de administración incorporada, pero es altamente extensible y permite una fácil integración con otras bibliotecas y módulos.
- Pyramid: Es flexible y altamente configurable, es adecuado para aplicaciones web de pequeña y gran escala. Proporciona un conjunto de recorridos de URL útiles predefinidos, vistas predefinidas y otras características útiles para los desarrolladores.
- Tornado: Tiene de alto rendimiento que es particularmente adecuado para encuestas largas, WebSockets y otros tipos de aplicaciones en tiempo real. [27]

Al tratar de usar los diferentes marcos de desarrollo el que presentó mayor cantidad de recursos y soportes fue Flask.

2.4.1.3 Flask

Flask es un marco de desarrollo web ligero para Python es adecuado para proyectos pequeños y medianos. Entre sus principales características tenemos:

- Flexible: Es muy flexible ya que permite fácil desarrollo de una amplia gama de aplicaciones web, que van desde aplicaciones simples de una pagina web a aplicaciones complejas con uso intensivo de datos.
- Ligero: Es muy liviano y rápido esto lo hace ideal para aplicaciones web de alto rendimiento que manejan gran cantidad de solicitudes.

- Simple: Permite que los desarrolladores se enfoquen principalmente en el desarrollo del código en lugar de tener que manejar gran infraestructura o código repetitivo que necesitan otros marcos de desarrollo.
- Gran comunidad: Cuenta con una comunidad grande y activa con gran cantidad de recursos, tutoriales y paquetes disponibles.
- Escalable: Se puede implementar en varias plataformas, se integra con otras herramientas como balanceadores de carga, sistemas de almacenamiento de cache y bases de datos lo que permite que maneje gran cantidad de tráfico y datos. [28]

Figura 2.27 Logo del marco de desarrollo web Flask.

Por estas características, pero principalmente por la gran cantidad de información, documentación, recursos y tutoriales Flask ha sido escogido como el marco de desarrollo web en el cual se ha desarrollado el proyecto.

2.4.2 Desarrollo de la aplicación web usando Flask

Para desarrollar una aplicación web usando Flask debemos contar con los requisitos que se listarán a continuación.

1. Un computador que cuente con un sistema operativo (Windows, Mac o Linux).
2. Editor de texto un Entorno Integrado de Programación (IDE). Que será usado para escribir el código fuente, algunos editores populares son Visual Studio Code, Pycharm o Sublime Text.
3. Python: Flask requiere que Python ya esté instalado en la computadora. Se puede descargar la última versión de Python a través del enlace oficial (https://www.python.org/).
4. Un navegador web que será usado para probar la aplicación web.

5. Instalar Flask que será necesario para crear la aplicación web. La instalación se usa al ejecutar el comando “pip install 40lask” desde la línea de comandos.

6. Librerías adicionales: Si se tiene funcionalidades adicionales es necesario instalar librerías, son necesarias si requerimos manejar formularios web, conexiones con una base de datos, manejar autenticación de los usuarios, prueba y depuración de la aplicación.

7. Servidor de alojamiento web: Esto consiste en la plataforma en la que será alojada nuestra aplicación.

8. Conocimiento básico de desarrollo web y tecnologías como HTML, CSS, JavaScript y HTML.

2.4.2.1 Instalación de requisitos principales

El desarrollo local de la aplicación web se realizará mediante una máquina virtual con sistema operativo Linux. Como se indicó en los pasos listados anteriormente primero debemos dirigirnos a la página web oficial de Python. En muchas versiones de Linux viene instalado Python por defecto. Una vez instalado comprobamos la versión instalada en nuestro sistema.

```
┌──(kali㉿kali)-[~]
└─$ python3 --version
Python 3.9.12
```

Figura 2.28 Versión instalada de Python.

El otro requisito principal es instalar Flask en nuestro sistema esto se logra a través del comando “pip install 40lask”

```
┌──(kali㉿kali)-[~]
└─$ pip install flask
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: flask in /usr/lib/python3/dist-packages (2.0.1)
```

Figura 2.29 Verificación de instalación de Flask.

Una vez instalado es necesario decidir en qué entorno se desarrollará la aplicación web. Se puede desarrollar a través de un simple editor de texto, pero un Entorno Integrado de Programación es adecuado para los requisitos del proyecto. El entorno escogido es Visual Studio Code.

Visual Studio Code se conoce como VSCode, es un Entorno Integrado de Programación sin costo de código abierto que creado por Microsoft. Cuenta con muchas herramientas y

funcionalidades entre ellas auto completado y navegación de código. Tiene opciones de depuración, interfaz altamente personalizable, integración con Git y soporta un gran número de lenguajes de programación. Es una de las opciones más populares al momento de escoger un entorno para desarrollar una aplicación. [29]

Para descargarla se debe ingresar a la dirección https://code.visualstudio.com/download y elegir la versión adecuada para nuestro sistema operativo.

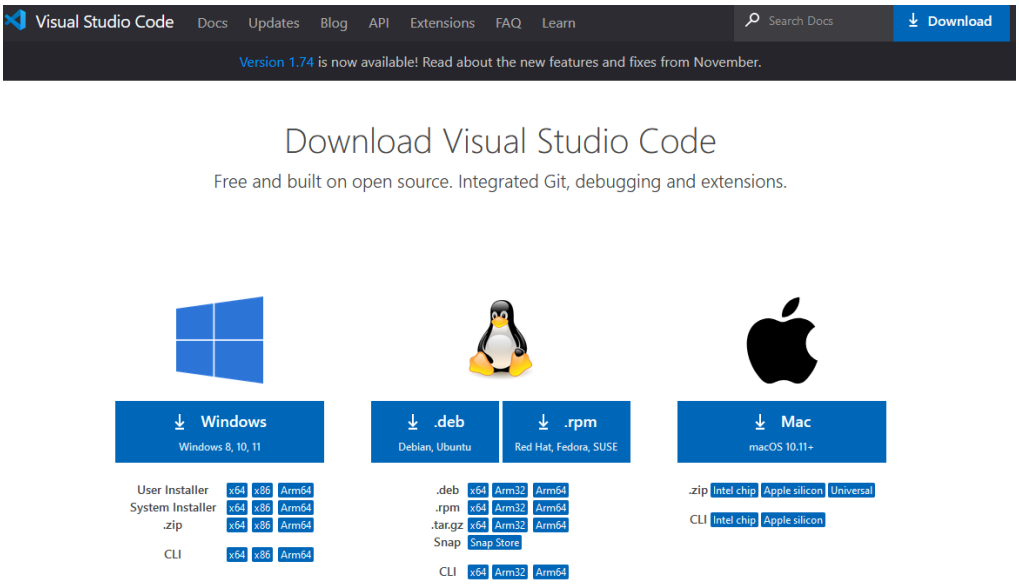


Figura 2.30 Página de descarga de Visual Studio Code

Una vez instalado nos encontraremos en la página principal de la aplicación.

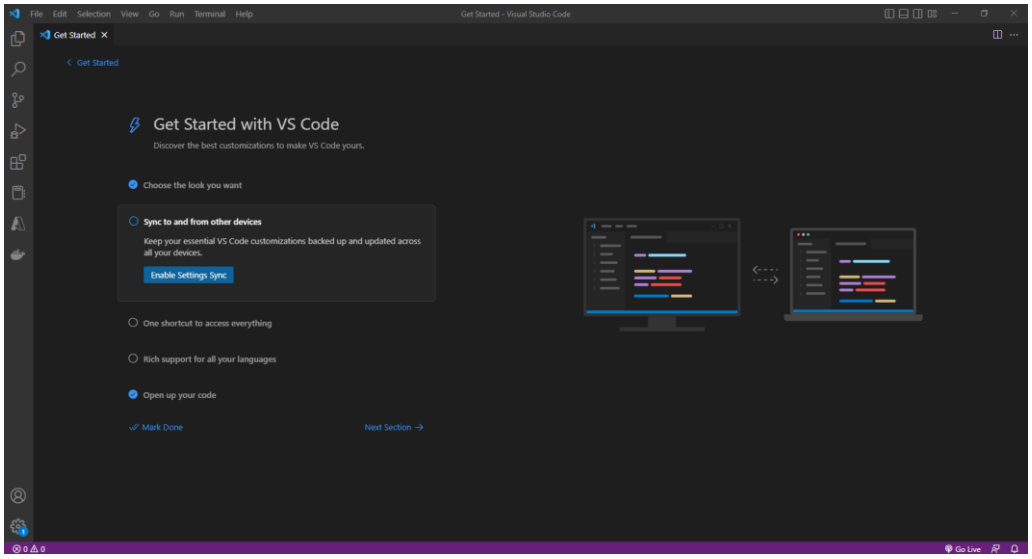


Figura 2.31 Pantalla de Bienvenida de Visual Studio Code.

2.4.2.2 Creación de una aplicación simple de prueba

Dentro de VSCode debemos elegir una carpeta para comenzar el desarrollo se recomienda la creación de una nueva carpeta. Debemos dar clic en File > Open Folder.

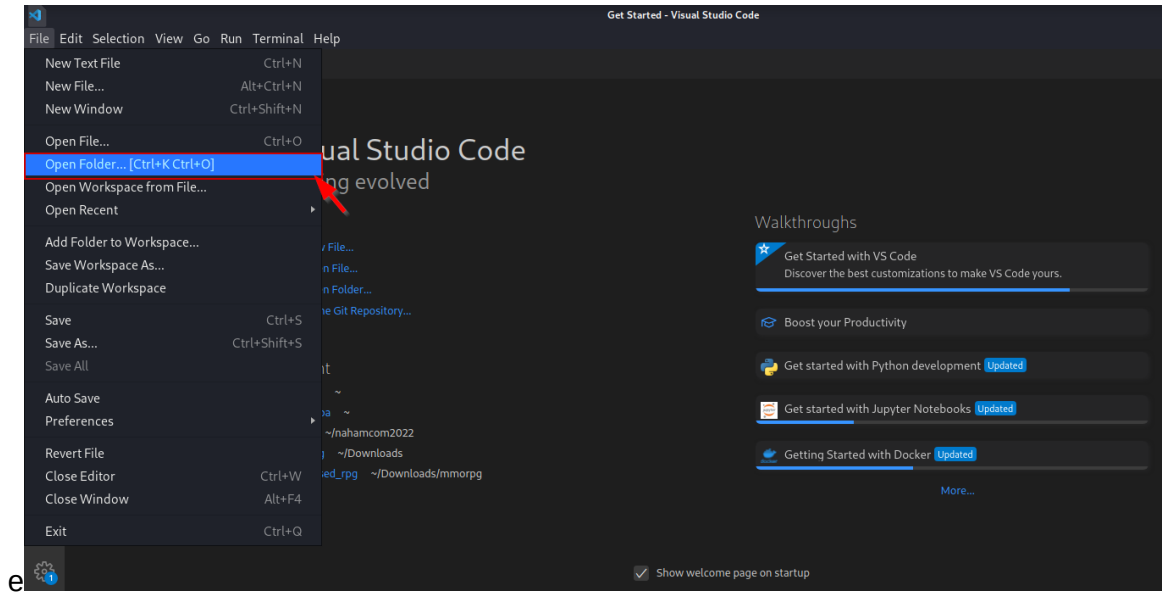


Figura 2.32 Pasos para abrir una carpeta.

Para crear un nuevo archivo debemos dar clic derecho el espacio que aparece debajo del nombre de nuestra carpeta y dar clic a New file

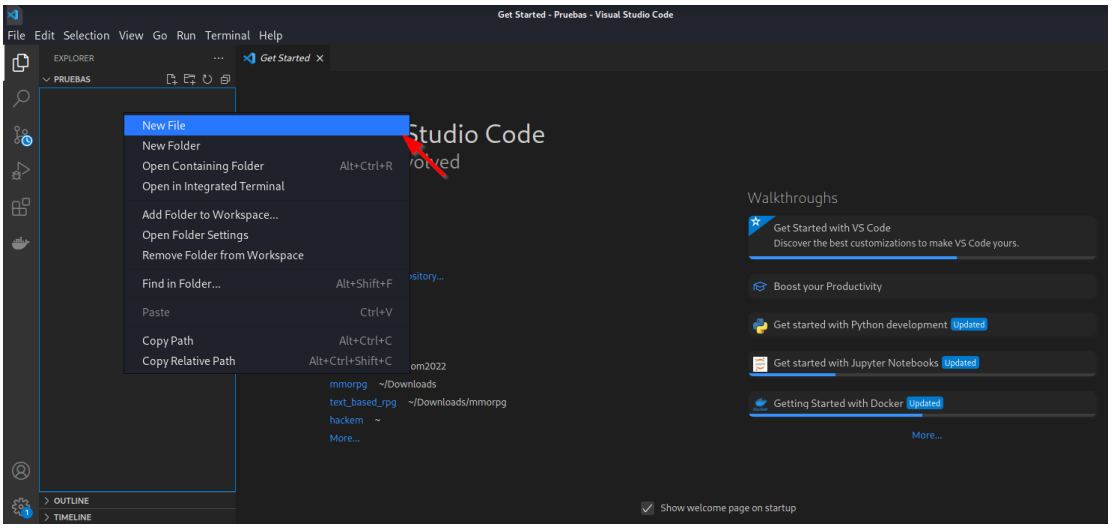


Figura 2.33 Creación de un nuevo archivo.

El nombre de nuestra simple aplicación será app.py dentro de esta usaremos el código provisto en la documentación de Flask para crear una simple aplicación. [30]

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def primera_prueba():
    return "<p>Primera aplicación</p>"
```

Script 2.3 Aplicación simple en Flask.

A continuación, en una terminal debemos llamar al módulo de Flask y ejecutar el siguiente comando:

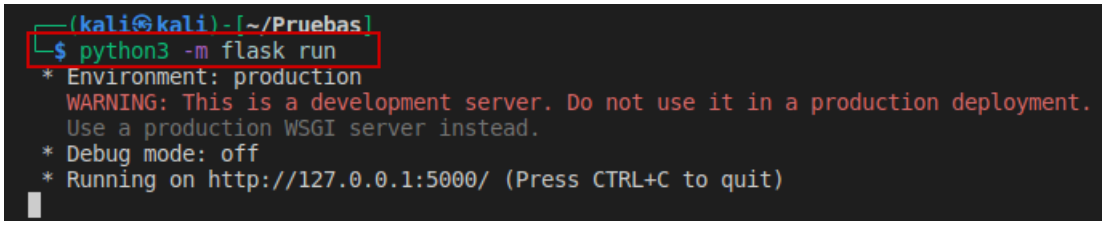


Figura 2.34 Ejecución del comando para correr la aplicación simple de manera local.

Nos informa que nuestra aplicación se encuentra corriendo en http://127.0.0.1:5000/ es decir en servidor local en el puerto 5000. Al ingresar a esa dirección tendremos:

Figura 2.35 Aplicación mostrada en el navegador web.

Como se puede observar en el navegador nuestra simple aplicación se encuentra funcionando de manera local. Todos los requisitos mínimos han sido cumplidos y podemos continuar con el desarrollo.

2.4.2.3 Creación del ambiente virtual de desarrollo

Nuestra aplicación será más compleja debido a que necesita ser desplegada en la nube, para esto debemos hacer uso de un ambiente de programación virtual. Aunque el ambiente de desarrollo no es estrictamente necesario es una buena práctica que facilitará el desarrollo.

El entorno virtual permite tener una versión específica de Python, además de versiones específicas de paquetes y bibliotecas para su proyecto esto lo aísla de otros proyectos que se estén desarrollado en la misma computadora. De esta manera se evita conflictos entre

diferentes versiones de paquetes y bibliotecas, con eso se facilita la administración e implementación de su aplicación.

Con el entorno virtual se puede administrar las dependencias y no preocuparse por actualizar o modificar una de las dependencias lo que puede frenar la aplicación, esto es de gran utilidad especialmente cuando trabaja en equipos o en diferentes computadoras. Muchos servicios de alojamiento web o plataformas en la nube, como Heroku, AWS y PythonAnywhere, esperan que las aplicaciones se implementen con sus dependencias aisladas en un entorno virtual. Por lo tanto, es mejor desarrollar con un entorno y facilitar la implementación y ejecución de su aplicación en diferentes plataformas. [31]

Para crear el ambiente virtual debemos ejecutar los siguientes comandos:

```
pip install virtualenv
virtualenv venv
source ./venv/bin/activate
```

Script 2.4 Instalación y creación de ambiente virtual.

Estos comandos son ingresados en la línea de comandos y así habremos creado nuestro ambiente virtual.

2.4.2.4 Paquetes usados en la aplicación web y funciones de Flask

La aplicación desarrollada debe considerar varias funcionalidades. Debe ser capaz de manejar solicitudes por parte del usuario y comunicarse con una base de datos. Se vio anteriormente que Flask por defecto no cuenta con un ORM quien permite la comunicación con la base de datos.

Object Relational Mapping (ORM) es una técnica usada para crear una especie de puente entre programación orientada a objetos y en muchos casos bases de datos relacionales. En bases de datos tradicionales un desarrollador debe escribir sentencias SQL para interactuar con una base de datos. Pero mediante un ORM los desarrolladores interactúan con la base de datos usando lenguajes de programación como en el caso de este proyecto Python y el ORM realizará la traducción entre comandos orientados a objetos y comandos SQL.

Un ORM también es una forma de abstracción ya que elimina la necesidad de cambiar el código fuente para interactuar con diferentes bases de datos como MySQL, PostgreSQL y SQLite.

Flask cuenta con una librería que le facilita el componente de ORM que recibe el nombre de SQLAlchemy es bastante robusto y flexible. [32]

La instalación de SQLAlchemy se la realiza a través de ‘pip install sqlalchemy’ en nuestro ambiente virtual. En nuestro código se debe encontrar las siguientes líneas para llamar a la librería e instanciar para poder usarlo.

```
from flask_sqlalchemy import SQLAlchemy
class Invernadero(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    content = db.Column(db.String(200), nullable=False)
```

Script 2.5 Llamada al paquete e instancia miento de SQLAlchemy.

Posteriormente se explicará los campos usados en la base de datos y su creación, en el código descrito se realizan pruebas de funcionamiento al crear las clases y algunos de los campos que tendrá la base de datos. Esto nos ayudará a determinar que el funcionamiento de manera local sea el adecuado.

Flask además cuenta con funciones que nos facilitan el desarrollo, entre las mas importantes tenemos:

- ‘Flask()’: Es la función constructora que crea la nueva instancia de la aplicación. Se usa al inicio
- ‘app.route()’: Esta función es usada para definir rutas en nuestra aplicación. Una ruta es un mapeo de una URL que maneja la lógica de dicha ruta. Esta función se usa para asignar direcciones URL.
- ‘render_template()’: Es una función usada para presentar una plantilla HTML y devolverla al usuario. Permite separar la lógica de la presentación, esto facilita el mantenimiento y la actualización de la apariencia de la aplicación.
- ‘request()’: Provee un objeto de solicitudes global que brinda acceso a los datos de la solicitud. Cuenta con múltiples utilidades para la manipulación de dichas solicitudes.
- ‘redirect()’: Se usa para redirigir el navegador del cliente a diferentes URL. Es de mucha utilidad para implementar funciones de ingreso y registro donde se debe enviar al usuario a otra pagina si han completado la acción satisfactoriamente.

```
from flask import Flask, render_template, url_for, request, redirect

@app.route('/', methods=['POST', 'GET'])
```

Script 2.6 Funciones de Flask y creación de una ruta que maneje métodos POST y GET.

2.4.2.5 Composición del proyecto en Flask

Un proyecto desarrollado en Flask tiene una estructura de archivos y carpetas que incluye los siguientes componentes:

1. Un archivo principal escrito en Python que define la instancia de la aplicación, define las rutas y la manera en que serán vistas. Se llama 'app.py'
2. Una carpeta 'templates' que es creada para que contenga las plantillas HTML que son utilizadas para mostrar las páginas. Usa un motor de plantillas como Jinja2 que permite insertar código de Python en las plantillas para generar contenido dinámico
3. Una carpeta 'static' que contiene los activos estáticos de la aplicación como son imágenes, hojas de estilo CSS y archivos JavaScript.
4. Un archivo 'requeriments.txt' que contiene las dependencias del proyecto y sus versiones. Esto hace que sea fácil instalar y ejecutar la aplicación en diferentes ambientes.
5. Adicionalmente se cuenta con carpetas llamadas 'models' y 'test', contienen modelos de la base de datos y unidades de prueba para la aplicación respectivamente. [33]

En el proyecto se realizará la conexión con una base de datos externa por lo tanto no es necesario. Por otro lado, la carpeta test no se la considera debido a que las rutas del proyecto no tienen gran complejidad.

```
proyecto_invernadero
│   app.py
│   requirements.txt
│   Procfile
└───templates
│   │   base.html
│   │   caratula.html
│   │   histórico.html
│   │   index.html
└───static
│   │   imagen1.jpg
│   │   imagen2.jpg
│   │   ...
│   │
│   └───css
│   │   │   main.css
│   │   │   ...
│   │
│   └───js
│       │   index.js
│       │   ...
```

Script 2.7 Diagrama que muestra la estructura de archivos y carpetas para el proyecto.

2.4.2.6 Archivos principales usados en el proyecto

Se ha hablado del archivo 'app.py' anteriormente, aquí se tiene el código en Python con las rutas, integraciones y funcionalidades del proyecto. Es el esqueleto de la aplicación que usa toda la estructura de la aplicación para brindarle funcionamiento y versatilidad. Por este motivo es un archivo que tiene una extensión considerable y puede ser encontrado en el Anexo I. Principalmente cuenta con las siguientes rutas y funciones.

```
from flask import Flask, render_template, url_for, request, redirect
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
class Invernadero(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    content = db.Column(db.String(200), nullable=False)

@app.route('/', methods=['POST', 'GET'])

@app.route('/caratula')

@app.route('/delete/<int:id>')

if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0")
```

Script 2.8 Estructura básica del archivo app.py

Como se muestra en el código se han creado tres rutas. La primera denotada por '/' es la raíz del proyecto y ahí se encuentra la parte principal de la aplicación donde se recibe las variables del invernadero y se hace el llamado a la base de datos para que muestre los datos históricos.

La siguiente ruta es '/caratula' aquí mostramos un mensaje de bienvenida del proyecto. Finalmente tenemos la ruta '/delete/<int:id>' esto se lo ha dejado para poder eliminar datos de la base de datos y se ha usado para ejecutar pruebas de funcionamiento. Finalmente, se cuenta con el sitio '/historico', aquí podremos ver todos los datos existentes en la base de datos.

El siguiente archivo usado en el proyecto corresponde a 'requeriments.txt' es un archivo de texto que enumera las dependencias de un proyecto de Python. Especifica los nombres de los paquetes y las versiones de las bibliotecas que el proyecto necesita para ejecutarse correctamente. Este archivo se utiliza para instalar los paquetes necesarios mediante pip, que es un instalador de paquetes para Python.

Para crear este archivo cuando se quiere ejecutar la aplicación y se han realizado cambios a los paquetes debemos ingresar el comando 'pip freeze > requirements.txt'. El archivo del proyecto contiene las siguientes librerías:

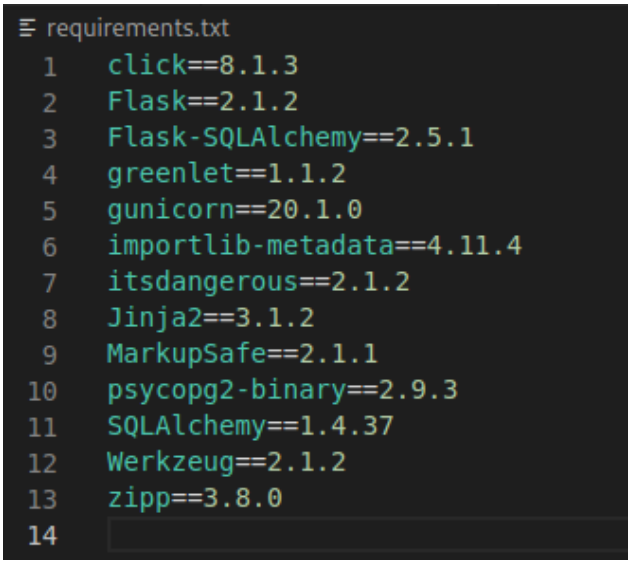


Figura 2.36 Contenidos del archivo requeriments.txt del proyecto.

Los siguientes archivos corresponden a la carpeta 'static' y contienen el código de las páginas estáticas del proyecto aquí encontramos las páginas HTML, los archivos de estilo CSS y JavaScript.

2.4.2.7 HTML

HTML (Hypertext Markup Language) es un lenguaje de marcado utilizado para crear sitios web. Es una combinación de etiquetas y atributos que se utilizan para estructurar, diseñar y definir el contenido de una página web.

Los documentos HTML se componen de elementos, que se representan mediante etiquetas. Estas etiquetas están encerradas entre corchetes angulares, como <etiqueta>. Un ejemplo básico de este tipo de archivos es:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Titulo elegido de la página web</title>
  </head>
  <body>
    <h1>Bienvenido a la primera prueba HTML</h1>
    <p>Aquí se incluyen los párrafos </p>
    <ul>
      <li>Item 1</li>
      <li>Item 2</li>
    </ul>
  </body>
</html>
```

Script 2.9 Estructura de un archivo HTML simple con elementos principales.

A continuación, se presentará una lista de las etiquetas principales en la creación de los archivos HTML.

Tabla 2.4 Tabla de etiquetas principales de HTML. [34]

Principales etiquetas	**Breve explicación**
!DOCTYPE	Versión empleada en la creación del archivo HTML
html	Define inicio y final para un archivo HTML
head	Indica información tipo meta del documento
title	Delimita el titulo del documento, se muestra en la barra de título del explorador
body	Contiene el contenido principal del documento
h1 – h6	Define las cabeceras y los números indican su tamaño
p	Define los párrafos
a	Define hipervínculos
img	Muestra una imagen dentro del documento
ul	Se usa para crear una lista sin orden
ol	Usado para crear listas con orden
li	Define una lista con ítems
table	Crea una tabla
tr	Indica la fila de una tabla
th	Indica celda cabecera de la tabla
td	Indica una celda de la tabla
form	Crea aun formulario que será usado para que los usuarios puedan ingresar datos
select	Crea una lista desplegable
option	Defina las opciones dentro de las listas desplegables

HTML es solo la estructura de una página web, no es un lenguaje de programación y no puede ser usado para la creación de funciones con interacción por sí mismo. En conjunto de CSS y JavaScript que son tecnologías que brindan dinamismo e interactividad se elaboran páginas web con estas características.

Flask usa un motor de plantillas, esto significa que al tener una plantilla llamada ‘base.html’ podemos escribir código que se repetirá en las páginas a las que invoquemos dicho código, esto permite que no se repita en cada una de las páginas elementos que ya ha sido creado en la base. Como ejemplo para nuestro proyecto tenemos:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/main.css') }}">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    {% block head %}
{% endblock %}
</head>
<body>
    {% block body %
}{% endblock %}
</body>
</html>
```

Script 2.10 Archivo base.html

Todo el código mostrado anteriormente será heredado y el nuevo código será incluido dentro de las sentencias '{% block head %}{% endblock %}'

El siguiente código es el empleado en la carátula del proyecto, aquí podemos observar cómo tenemos una sentencia que indica que extendemos de la plantilla 'base.html' y escribimos el código entre las sentencias '{% block head %}{% endblock %}'.

```
{% extends 'base.html' %}

{% block head %}
<title>EPN</title>
<style>
    body {
      background-image: url('/static/img/pexels-johannes-plenio-
1423601.jpg');
      background-size: cover;
      background-repeat: no-repeat;
    }
  </style>
{% endblock %}

{% block body %}

<div style="display: flex; justify-content: center; align-items:
center; height: 100vh;">
    <div style="width: 55%; text-align: center;">
      <!-- Your text content goes here -->
      <h1> Escuela Politécnica Nacional</h1>
      <h1>Facultad de Ingeniería Eléctrica y Electrónica</h1>
      <h1>Trabajo de Integración Curricular</h1>
      <h2>Estudio, diseño e implementación de una solución para la
recolección de datos provenientes del sistema de riego usando servicios
en la nube y presentación de los datos mediante una página web</h2>
      <h3>Estudiante: Jonathan Alvarez</h3>
      <h3>Director: Ramiro Morejón</h3>
      <div style="text-align: center; margin-top: 20px;">
```

```
        <a href="/"><button style="font-size: 25px; padding: 10px 20px;
background-color: #1f6fb1; color: white; border: 2px solid rgb(1, 7,
17); font-weight: bold;border-radius: 15px;">Ingresar</button></a>
      </div>
    </div>
    <div style="width: 45%; text-align: center;">
      <!-- Your image goes here -->
      <img src="/static/img/epn.png" alt="Logo Escuela Politecnica
Nacional" width="50%" height="50%">
    </div>
  </div>

{% endblock %}
```

Script 2.11 Código HTML del archivo caratula.html

Una estructura similar tiene el archivo principal del proyecto, por su amplio contenido se lo puede encontrar en el Anexo II.

2.4.2.8 CSS

Cascading Style Sheet (CSS) traducido como hojas de estilo en cascada es un lenguaje utilizado en la descripción de la presentación en documentos elaborados en HTML. Se utiliza para controlar el diseño, los colores, la fuente y otros elementos visuales de las páginas web. CSS permite separar la presentación de la página web de su estructura definida por HTML.

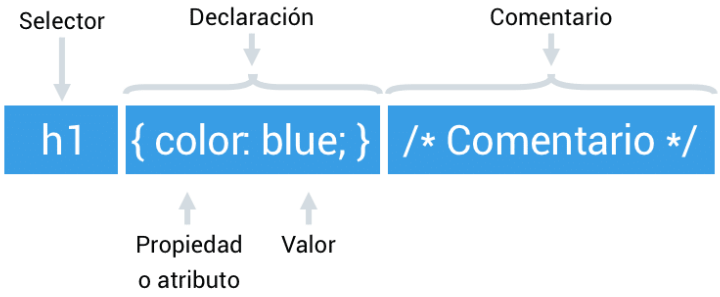


Figura 2.37 Sintaxis de selector CSS.

Al usar CSS, es posible cambiar la apariencia de todos los elementos en una página web modificando un solo archivo en lugar de cambiar la apariencia de cada elemento individualmente. Los estilos CSS se definen mediante selectores, que seleccionan los elementos HTML a los que se deben aplicar los estilos, y un conjunto de reglas que definen los estilos que se aplicarán.

A continuación, se muestra un ejemplo de un archivo CSS simple:

```
body {
    background-color: lightgray;
```

```
}

h1 {
    color: brown;
    text-align: right;
}

p {
    font-family: Helvetic;
    font-size: 15px;
}
```

Script 2.12 Archivo CSS simple.

Existen tres formas de incluir CSS en la página web: dentro de la etiqueta HTML, usando un archivo externo o incluir en el documento en la sección head. Entre las principales características de CSS se encuentran:

- Selectores y reglas: Brindan la posibilidad de seleccionar elementos HTML y aplicar las reglas de estilo a estos. Incluyen las propiedades como color de fondo, color de texto, tipo de letra, entre otros.
- Propiedades: Describen como se presentan los elementos seleccionados. Las propiedades mas comunes son: color, alto, tipo de letra, ancho, margen, borde, etc.
- Valores: A las propiedades se asignan valores para definir como representar los elementos. Un ejemplo es la asignación a color en notación RGB o hexadecimal.
- Herencia: Las propiedades se heredan de elementos padres a hijos, esto quiere decir que los estilos de un elemento padre se aplican por defecto a los elementos hijos.

CSS cuenta con un gran número de propiedades, los más importantes serán mostrados en la siguiente tabla.

Tabla 2.5 Atributos y propiedades principales de CSS. [35]

Propiedades	**Descripción**
color	Especifica el color de un texto
font-family	Indica el tipo de fuente que será usado en el texto
font-size	Indica el tamaño del texto
text-align	Sirve para indicar la alineación horizontal del texto
padding	Añade espacio dentro de un elemento
margin	Añade margen fuera del elemento
border	Añade borde alrededor del elemento

width	Indica el ancho de un elemento
height	Indica el alto de un elemento
background-color	Sirve para especificar el color de fondo de un elemento
display	Especifica como un elemento debe ser mostrado
position	Especifica el método de posicionamiento usado para el elemento

El archivo de CSS para el proyecto tiene el nombre de 'main.css' cuenta con la estructura mencionada anteriormente y muchos de los atributos mostrados en la tabla.

2.4.2.9 JavaScript

La última tecnología principal usada en el proyecto es JavaScript. Este lenguaje de programación es usado para brindar a los sitios web capacidad de ser interactivos y dinámicos. Brinda la capacidad de crear características complejas en las páginas web. Es un lenguaje del lado del cliente, esto quiere decir que se ejecuta en el navegador del usuario. [36] JavaScript permite a los desarrolladores crear múltiples elementos interactivos que van desde controles deslizantes para imágenes, validación de formularios, mapas interactivos para páginas web y en el caso de este proyecto la creación del gráfico en el tiempo para las variables recibidas del sistema de invernaderos.

Mediante los datos recibidos se ha propuesto crear un gráfico que permita observar su variación en el tiempo, para esto se ha hecho uso de una librería llamada ChartJS. Permite crear de manera fácil diferentes tipos de gráficos usando elementos del lienzo en HTML5. ChartJS cuenta con una configuración por defecto que permite implementar gráficos de manera sencilla, no será necesario modificar opciones para que se desplieguen sin problema y cuenta con animaciones por defecto. [37]

El siguiente código pertenece al gráfico creado usando ChartJS. Se observa que en primera instancia debemos usar la etiqueta "div" y usar la etiqueta "canvas" junto con un identificador, aquí se define en que parte de la página web aparecerá el gráfico. Dentro de la etiqueta "script" primero debemos usar la fuente de la librería, es decir dónde se encuentra librería hospedada. La siguiente etiqueta contiene la instancia del objeto junto a las características necesarias, que son su nombre y el tipo de gráfico. Luego creamos propiamente la gráfica usando la integración con Flask que nos permitirá acceder a los datos dentro de la base de datos para que sean representados.

```
<div>
  <canvas id="Chart"></canvas>
</div>
```

```
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<script>
        var ctx = document.getElementById("miTabla").getContext("2d")
        var lineChart = new Chart(ctx, {
                type: "line",
                data: {
                        labels: {{ fech | tojson }},
        datasets: [
                {
                        label: "Variable en el ultimo dia ",
                        data: {{ var | tojson }}
                                        }
        ]
                        },
        options: { }
                })

</script>
```

Script 0.13 Implementación de ChartJS.

2.5 Despliegue de la aplicación, creación de la base de datos en la nube y envío de datos localmente

Desplegar la aplicación web en el servicio en la nube es hacer que la aplicación web esté disponible a los usuarios a través de internet. Este proceso incluye la configuración del servidor, preparar los archivos de la aplicación y configurar la infraestructura necesaria.

Al haber seleccionado una Plataforma como Servicio (PaaS) para hospedar la aplicación el proceso del despliegue se ve simplificado. Esto se debe a que la plataforma se encarga de la infraestructura necesaria y administración del servidor. Así, es necesario enfocarse en preparar la aplicación junto a sus archivos y subirlos hacia el proveedor de la plataforma. Además, la plataforma se encargará de otros aspectos como configuración del servidor, escalado y mantenimiento.

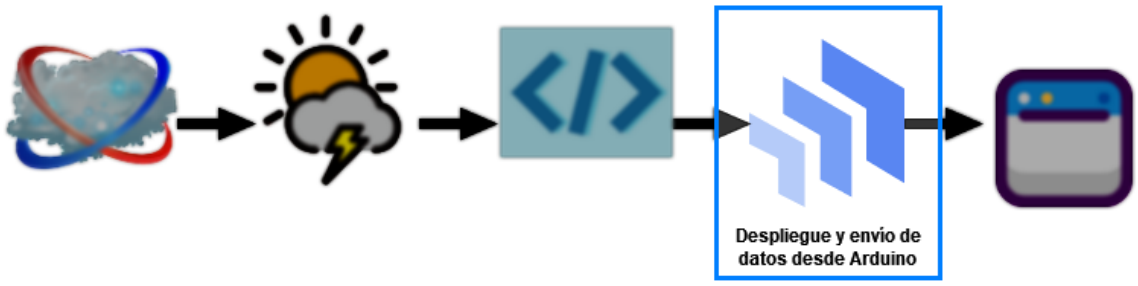


Figura 2.38 Etapa 4: Despliegue de la aplicación y envío de datos desde el Arduino Yun.

2.5.1 Despliegue de la aplicación en Heroku

La plataforma seleccionada para el despliegue es Heroku, permite alojar los archivos del proyecto dentro de su infraestructura y provee una dirección web que permite acceder a la aplicación desde cualquier dispositivo conectado a internet.

Heroku cuenta con numerosas integraciones con múltiples como son: Git, GitHub, Hashicorp Terraform, etc. En este proyecto el despliegue se lo realiza a través de la herramienta Git.

2.5.1.1 Herramienta para el control de versiones de sistema Git

Git es una herramienta libre con código abierto para realizar Control de Versión de Sistema (VCS) distribuido. Permite a desarrolladores seguir y administrar cambios realizados al código fuente de los programas, controlar las diferentes versiones del código, múltiples miembros del equipo trabajar en el mismo código y fusionar el código a la base del mismo sin que existan conflictos. Además, permite volverá versiones anteriores fácilmente.

Cuando existen diferentes líneas en el desarrollo Git las conoce como ramas. Git anima a los desarrolladores a tener múltiples ramas locales que puedan ser independientes unas a las otras. Al crear un repositorio Git este tiene una rama para el código base llamada “master”, una vez los cambios han sido realizados y probados en las ramas secundarias se pueden fusionar al código base. Esto permite mantener limpio, organizado y limpio al código fuente de los programas. [38]

2.5.1.2 Prerrequisitos del despliegue en Heroku

Para el despliegue en Heroku es necesario contar con Git y Heroku CLI. Git se encuentra instalado por defecto en muchas distribuciones Linux como es el caso de donde se realizó el proyecto de manera local. Para comprobar si se encuentra instalado ingresamos el comando ‘git --version’.

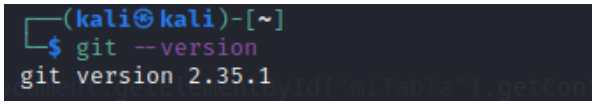


Figura 2.39 Git instalado en la máquina donde se desarrolla el proyecto de manera local.

Heroku cuenta con amplia documentación para la instalación de Heroku CLI. Existen múltiples opciones para instalación dependiendo del sistema operativo, la correspondiente al sistema usado consiste en ejecutar el comando ‘curl https://cli-assets.heroku.com/install-

ubuntu.sh | sh'. Una vez ejecutado podremos verificar la versión instalada en nuestro sistema usando 'heroku --version'

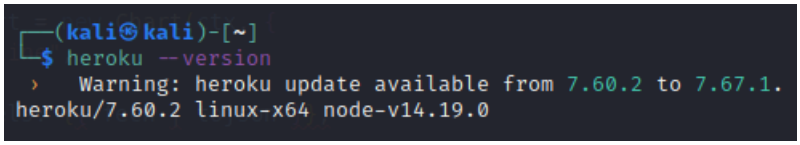


Figura 2.40 Versión de Heroku CLI instalada.

El siguiente paso consistirá en ingresar a Heroku usando el comando 'heroku login'. Un texto nos indicará que debemos presionar cualquier botón para ser redirigidos a una pagina web e ingresar nuestros datos de registro.

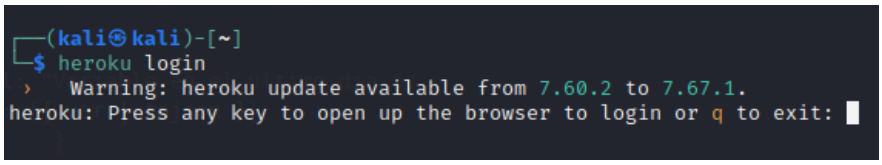


Figura 2.41 Ejecución del comando que nos redirigirá a la página web para ingresar las credenciales.

Una vez completados estos requisitos previos será posible proceder con el despliegue de la aplicación.

2.5.1.3 Despliegue mediante Git

Una vez instalado Git y Heroku CLI es posible desplegar la aplicación mediante línea de comandos en la consola. El comando 'heroku create' crea una aplicación vacía en Heroku junto a un repositorio Git remoto vacío, este será vinculado al que disponga de manera local el proyecto.

Figura 2.42 Creación de la aplicación y repositorio Git remoto.

El siguiente paso consiste en inicializar el repositorio a través del comando 'git init' esto dentro de la carpeta del proyecto seguido del comando 'git add .' y 'git commit 'm "Primera versión"'. Los comentarios nos permitirán describir los cambios para poder identificar las versiones del proyecto.

```
┌──(env)─(kali㉿kali)-[~/invernadero]
└─$ git commit -m "Cambios de finales de prueba"
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   templates/base.html
```

Figura 2.43 Ejecución del comando commit para los cambios realizados.

Finalmente, para desplegar el código en Heroku se usa el comando ‘git push’ que enviará el código desde el repositorio local hacia la rama principal del proyecto en los servidores de la plataforma.

```
┌──(env)─(kali㉿kali)-[~/invernadero]
└─$ git push heroku main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 6 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 930 bytes | 930.00 KiB/s, done.
Total 5 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-22 stack
remote: -----> Using buildpack: heroku/python
remote: -----> Python app detected
remote: -----> No Python version was specified. Using the same version as the last build: python-3.10.9
remote:        To use a different version, see: https://devcenter.heroku.com/articles/python-runtimes
remote: -----> No change in requirements detected, installing from cache
remote: -----> Using cached install of python-3.10.9
remote: -----> Installing pip 22.3.1, setuptools 63.4.3 and wheel 0.37.1
remote: -----> Installing SQLite3
remote: -----> Installing requirements with pip
remote: -----> Discovering process types
remote:        Procfile declares types -> web
remote:
remote: -----> Compressing...
remote:        Done: 31M
remote: -----> Launching...
remote:        Released v6
remote:        https://invernadero-epn.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/invernadero-epn.git
   b75ab2c..fed016d  main -> main
```

Figura 2.44 Despliegue de la aplicación en Heroku.

Como se puede observar los archivos locales proceden a ser enviados al repositorio remoto. Además, debe incluirse un archivo llamado ‘Procfile’ este documento le indica a Heroku como debe ser ejecutada la aplicación, para este proyecto se debe incluir ‘web: gunicorn app: app’. El comando especifica las sentencias que deben ser ingresadas para encender el servidor web de la aplicación usando un marco de trabajo llamado Gunicorn.

Gunicorn es básicamente un servidor web WSGI, este tipo de servidores sirven de interfaces entre una puerta de entrada en este caso WSGI y la aplicación web permitiéndole mantenerse estable. Es la opción mas confiable si no se esta seguro que tipo de WSGI usar. [39]

2.5.2 Creación de la base de datos en la nube

Como se vio anteriormente Azure es una plataforma de servicios en la nube que provee una gran variedad de servicios los cuales incluyen bases de datos. Esta plataforma fue elegida para hospedar la base de datos debido a que de igual manera provee un nivel de servicio estudiantil con crédito disponible 100$ para su uso durante un año.

Azure dispone de gran cantidad de servicios relacionados con bases de datos algunos de ellos son: Azure SQL Database, Azure Cosmos DB, Azure Database para MySQL, Azure Database for PostgreSQL, Azure Database for MariaDB. Todos estos ofrecen bases de datos completamente administrados, esto significa que Azure se encarga de la infraestructura necesaria, además brinda respaldos, actualizaciones y escalabilidad. Así el desarrollador puede enfocarse en su aplicación y los datos en lugar de preocuparse por la infraestructura.

2.5.2.1 PostgreSQL

Flask por defecto no ofrece un ORM, esto permite al desarrollador escoger el tipo de base de datos que crea necesaria para el proyecto. PostgreSQL es una base de datos de relacional orientada a objetos y administrada de código abierto. Es conocida por ser robusta, confiable y capaz de manejar gran cantidad de datos además de usuarios concurrentes. Es ampliamente usada en desarrollo web, almacenamiento de datos e inteligencia de empresas, se puede usar para otros fines como registro de datos científicos y registro de usuarios. [40]

Debido a que Flask y PostgreSQL son de código abierto comúnmente se ven juntos en muchas aplicaciones. De igual manera, en distinta documentación y tutoriales se encuentran guías para la integración de ambas tecnologías. Debido a que Flask no cuenta con un ORM que le permita comunicarse directamente se usa la librería SQLAlchemy que permite comunicación muy sencilla de la base de datos y la aplicación.

Figura 2.45 Logo de PostgreSQL.

2.5.2.2 Creación de la base de datos en Azure

A continuación, se mostrarán los pasos para crear una base de datos tipo PostgreSQL usando Microsoft Azure.

1. Ingresar a la cuenta en el portal de Azure https://portal.azure.com/.
2. Dar clic en el símbolo “+ Crear recurso”.

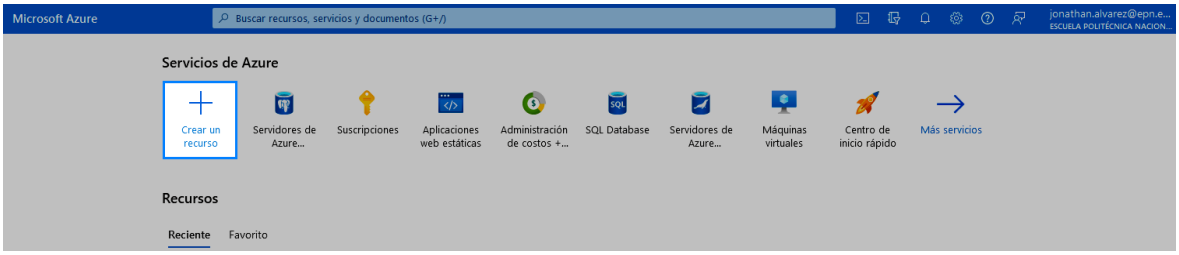


Figura 2.46 Crear un nuevo recurso en Microsoft Azure.

3. En la caja de búsqueda ingresamos “PostgreSQL”. Y elegir la opción PostgreSQL Server.

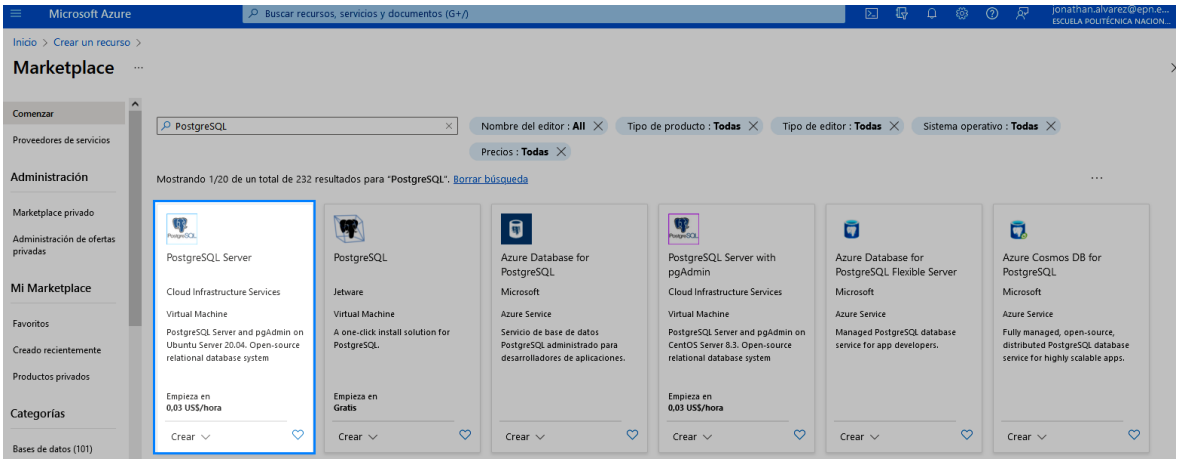


Figura 2.47 Selección del servicio PostgreSQL Server.

4. En la ventana que se muestra se elige el plan y dar clic a crear.

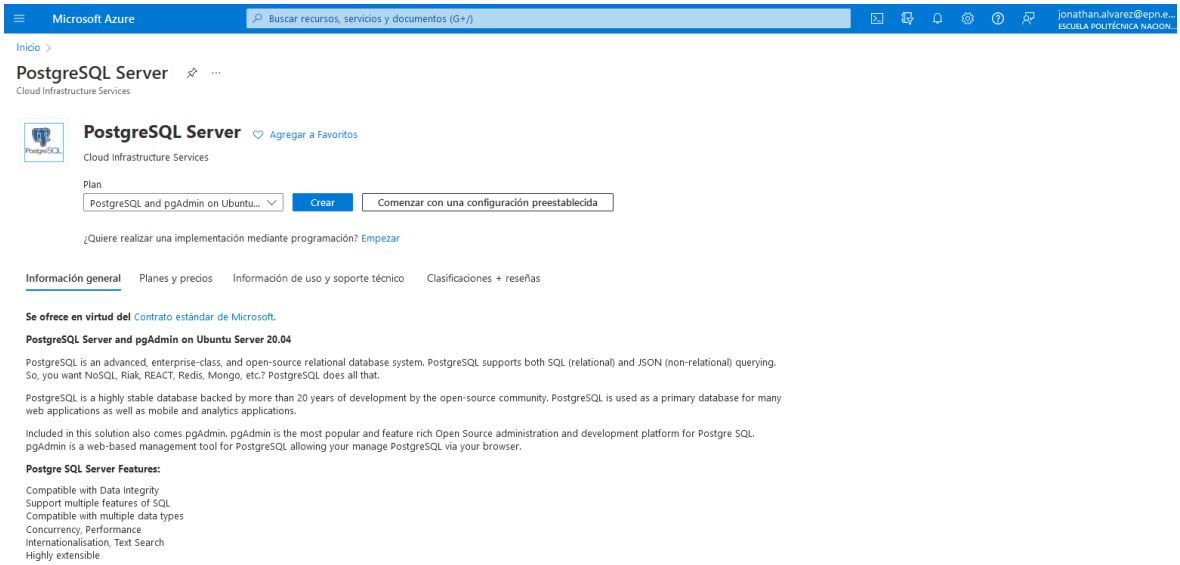


Figura 2.48 Selección del tipo de plan y creación de la base de datos.

5. Llenamos los datos necesarios, primero la suscripción para este se debe seleccionar Azure para estudiantes. Se debe crear un nuevo grupo de recursos que es un conjunto de servicios que estarán bajo esa categoría. Luego nombre de instancia y región, en región es recomendable East US. Y seleccionar las opciones de imagen para la instalación.

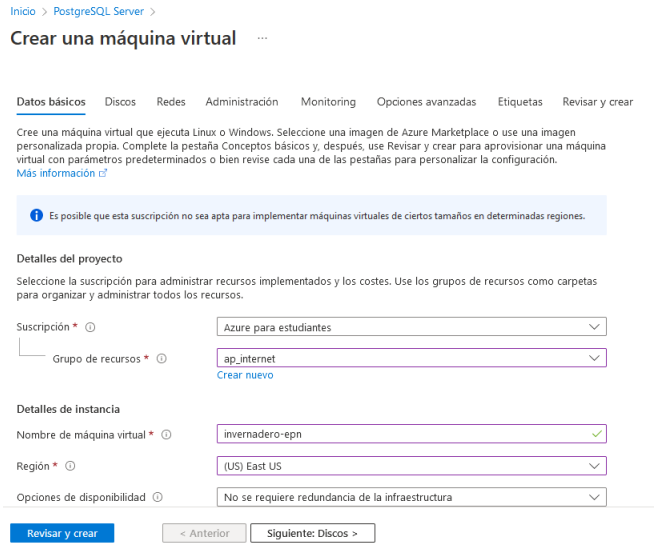


Figura 2.49 Primeros pasos para la creación de la máquina virtual.

En las siguientes opciones se deja las opciones por defecto, en la opción de tamaño existen opciones que son más económicas. En la parte final seleccionar la opción de contraseña para la operación de la base de datos para poder acceder a través de herramientas.

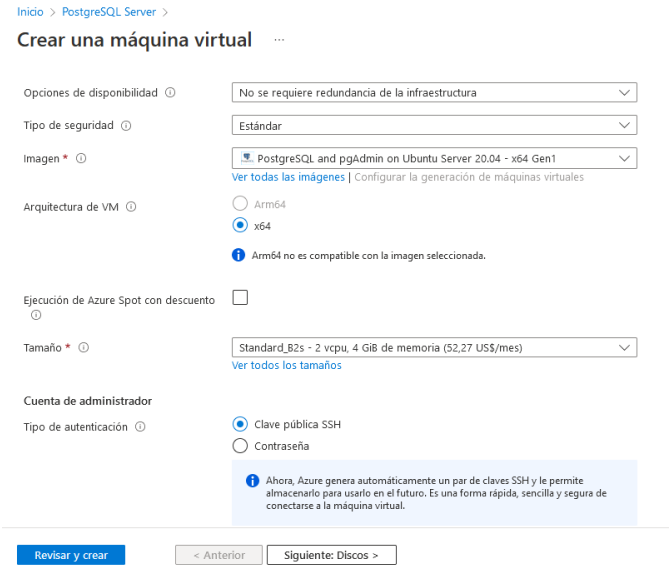


Figura 2.50 Opciones de imagen de la base de datos y tipo de máquina.

6. Seleccionar el tipo de maquina en el cual se ejecutará la base de datos. Para que la base de datos sea almacenada es necesario una máquina virtual, debido a que el proyecto no requiere gran cantidad de datos se seleccionó la opción mas económica.

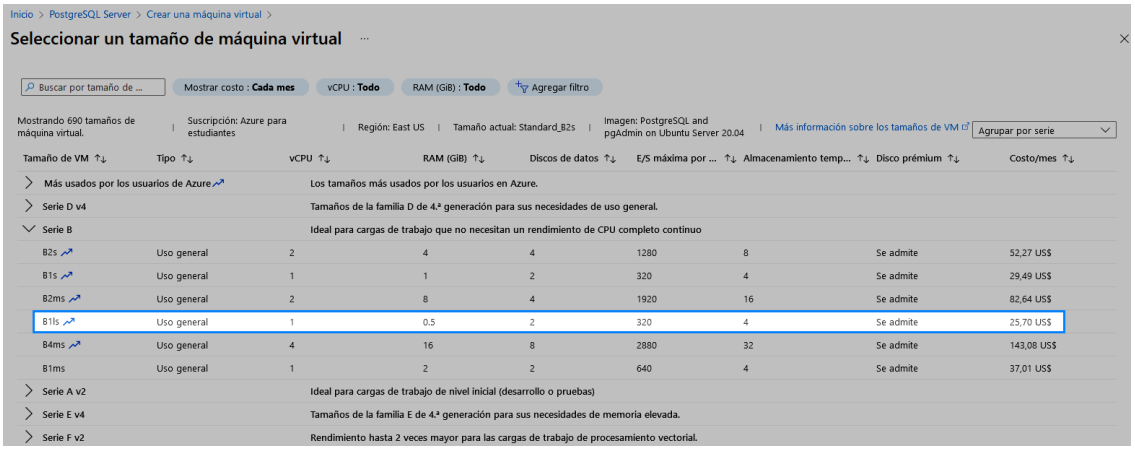


Figura 2.51 Selección del tipo de máquina virtual.

7. Seleccionar el tipo de disco que tendrá la máquina virtual. Para el proyecto debido a que no usará mucho espacio ni será intensivo en carga una buena opción es usar un disco duro mecánico HDD.

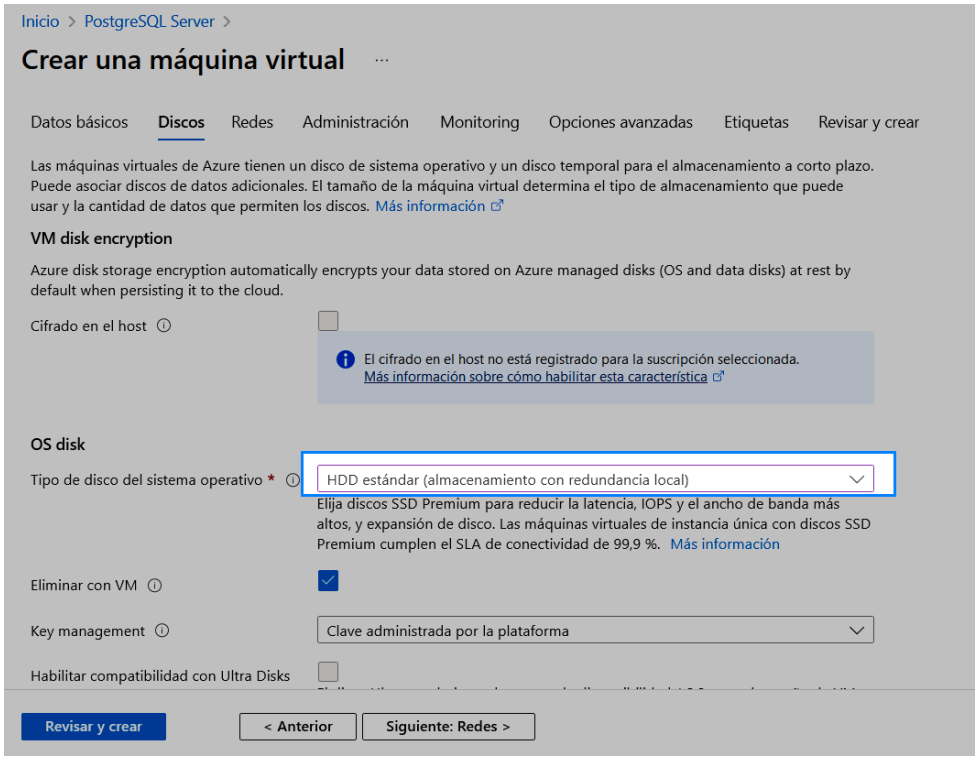


Figura 2.52 Selección de tipo de disco duro para la máquina virtual.

8. Seleccionar las opciones de red. En este paso no se realizan mayores modificaciones a las opciones por defecto.

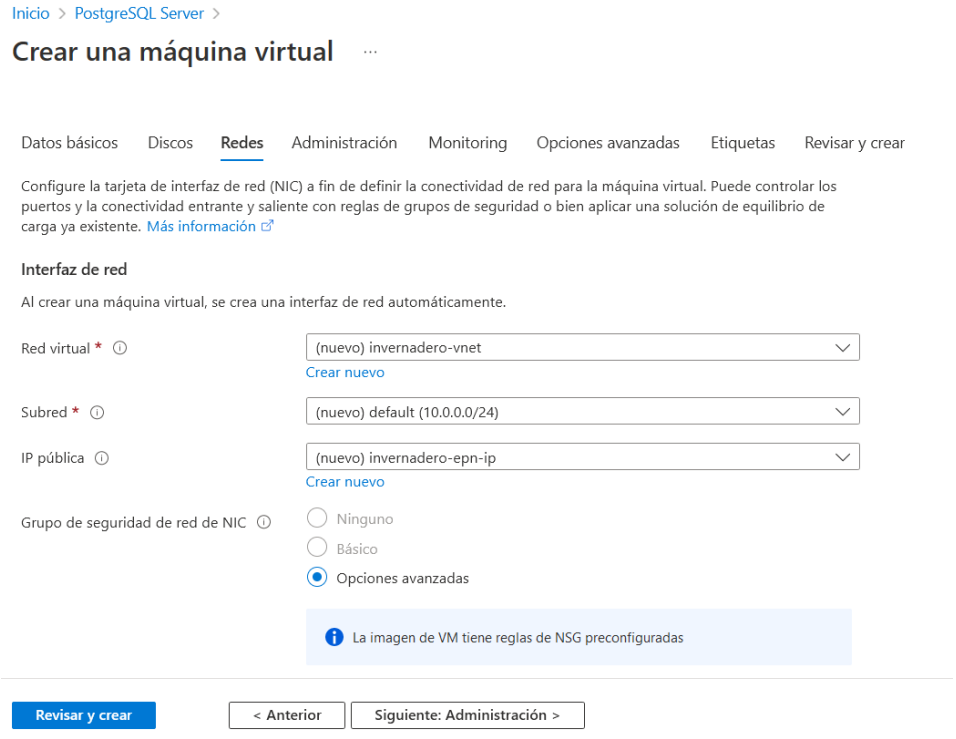


Figura 2.53 Selección de las opciones de redes.

9. Las siguientes opciones se las deja por defecto, continuar hasta llegar a revisar y crear. En la pestaña final se tendrá un resumen de las opciones escogidas.

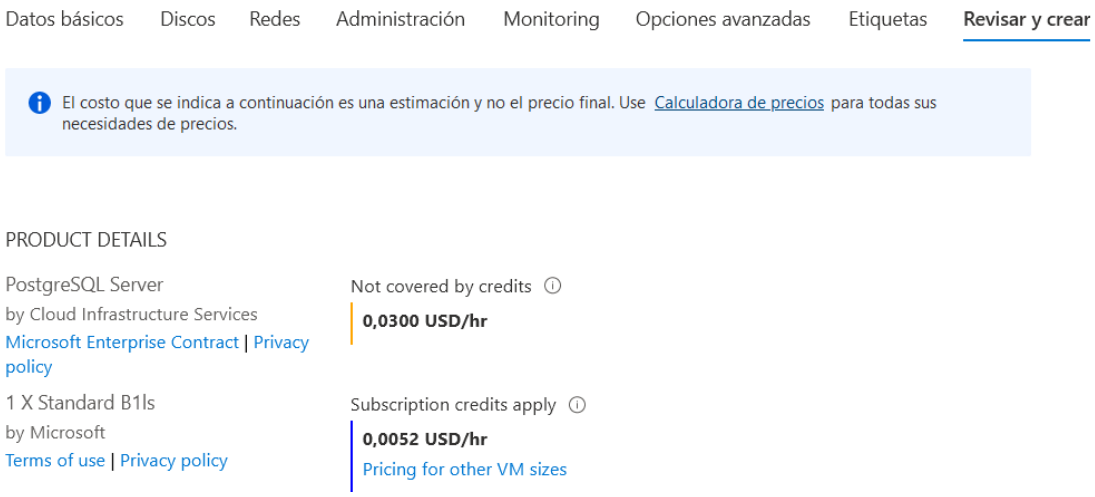


Figura 2.54 Resumen del costo de la máquina virtual.

Aquí se muestra el costo que tendrá el uso de la máquina virtual sobre la cual el servidor administrado de PostgreSQL se encontrará almacenado. Gracias al crédito que provee la cuenta institucional se puede acceder a este tipo de servicios.

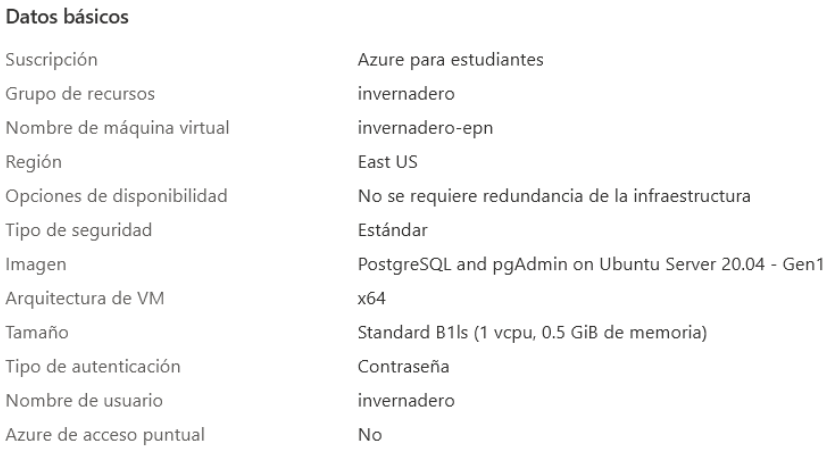


Figura 2.55 Resumen completo de la creación de la máquina virtual.

Es importante revisar que las opciones elegidas sean las adecuadas para el proyecto, al tener una plataforma con costo la empresa suele recomendar configuraciones que tienen un precio extra. Microsoft Azure permite posteriormente escalar los servicios de ser necesario, muchas de las configuraciones se pueden cambiar posteriormente de ser necesario.

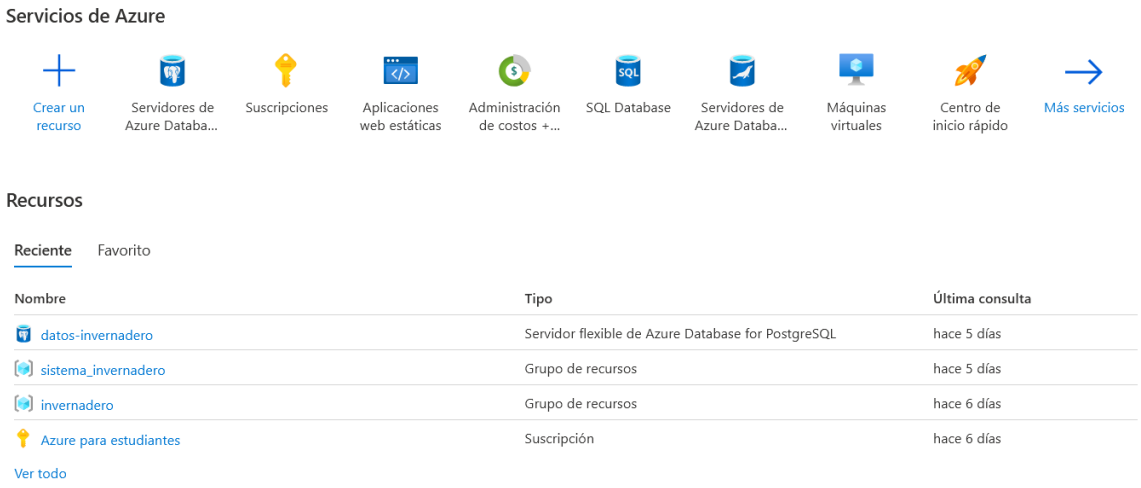


Figura 2.56 Portal de Azure con los recursos creados.

Si damos clic sobre el recurso de tipo Servidor flexible de Azure de base de datos PostgresSQL se mostrará la ventana de administración. En esta ventana se tiene información general de la base de datos y opciones de configuración. Además, cuenta con gráficos de uso y de carga que permiten determinar si los recursos elegidos son los apropiados o existe la necesidad de escalar la base de datos para que tenga mejor desempeño.

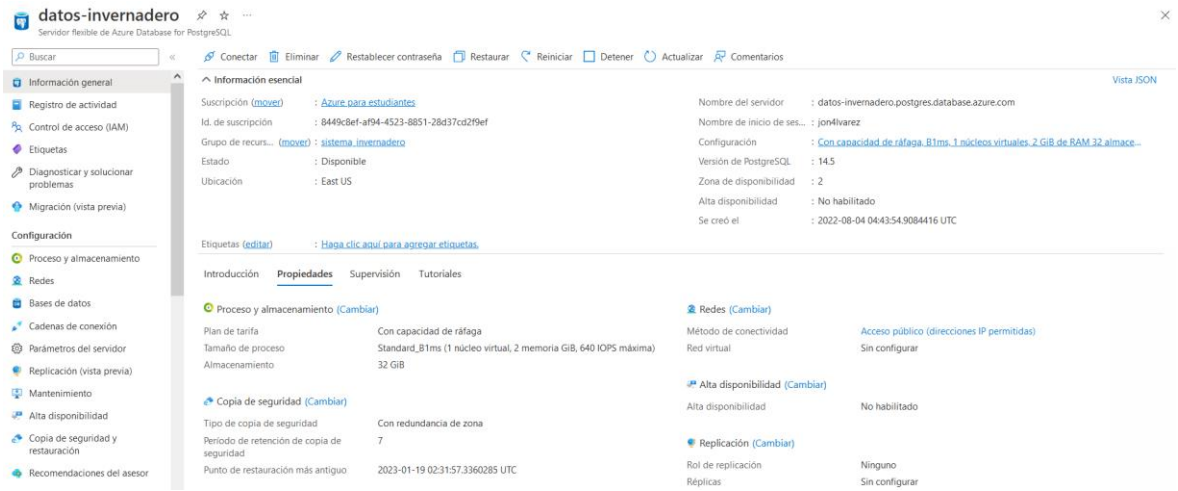


Figura 2.57 Panel de administración de la base de datos.

En la parte izquierda se encuentran todas las opciones que contiene la base de datos para su configuración, si se desea escalar o cambiar parámetros con la seguridad en este panel se encontrará la manera de hacerlo. En la parte inferior se encuentran opciones de supervisión, automatización y soporte, que se las debe tener en cuenta en caso de ser necesarias.

2.5.2.3 Comunicación entre la base de datos y la aplicación web

Al tener la base de datos desplegada con el servicio de Microsoft Azure y la aplicación web con Heroku que son dos proveedores se debe encontrar una manera de vincular ambos servicios para completar la función del proyecto. La aplicación web debe encargarse de hacer solicitudes que la base de datos debe responder, de esto se encarga el ORM que traduce las sentencias que se encuentran en el programa al idioma que maneja la base de datos. Para que esta tarea se lleve a cabo ambos elementos deben estar vinculados a través de una cadena de conexión.

La cadena de conexión es una cadena de caracteres que contienen información que especifica como conectarse a la fuente de los datos que en este caso es una base de datos. Incluye información como el nombre del servidor, puerto, nombre de la base de datos y credenciales para establecer la comunicación [41]. A continuación, se muestra un ejemplo de cadena de conexión con sus características.

postgresql://usuario:contraseña@servidor:puerto/basededatos

- 'postgresql': Indica el tipo de base de datos.
- 'usuario:contraseña': Son las credenciales para autenticarse en la base de datos.
- 'servidor': Define la localización del servidor donde se encuentra corriendo la base de datos.
- 'puerto': Número del puerto en el cual el servidor está a la escucha de conexiones.
- 'base de datos': El nombre de la base de datos a la que se requiere conexión.

Es de gran importancia mantener esta cadena de conexión segura y no compartirla de manera abierta debido a que contiene información sensible como las credenciales y la localización del servidor.

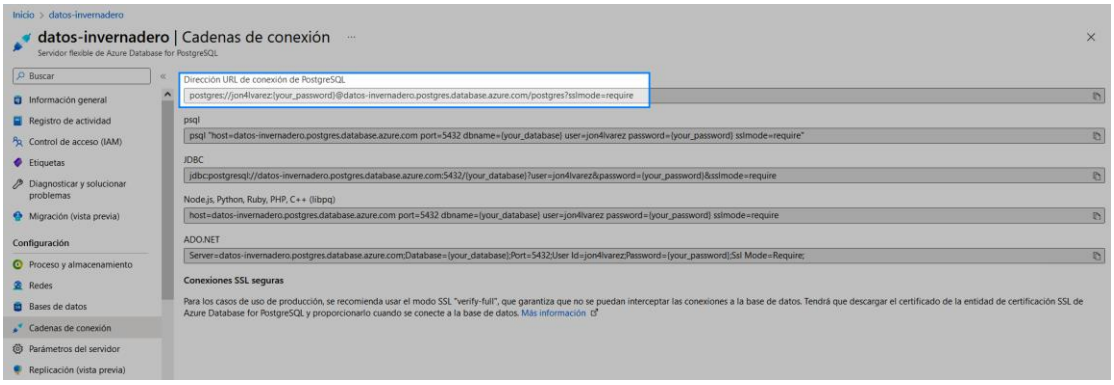


Figura 2.58 Cadenas de conexión proporcionadas por Azure.

Como se puede observar se muestra toda la información relevante para la conexión, por razones de seguridad la cadena mostrada no cuenta con la contraseña colocada. Al momento de realizar el llamado a la base de datos en la aplicación web se debe colocar la contraseña en la cadena y la conexión será posible.

2.5.3 Envío de datos de manera local

Para las pruebas preliminares de la aplicación el despliegue local se encuentra conectado a la base de datos en la nube. La simulación del comportamiento del Arduino Yun con la base de datos se la realizó a través de un script realizado en Python que permite hacer peticiones a la aplicación de la misma manera que el Arduino las llevaría a cabo.

```
import requests
from sympy import content
import random

# URL de la aplicacion
# https://invernadero-epn.herokuapp.com/

requests.post(url="http://localhost:5000",
    data={
        "content":"Localizacion invernadero",
        "temperatura":random.randint(0, 13)+28,
        "humedad":random.randint(0, 35)+54,
        "humsuelo":random.randint(0, 20)+40,
        "nivelPH":random.randint(0, 2)+6,
        "luminosidad":random.randint(0, 99)+1
    })
```

Script 2.14 Simulación de peticiones.

De igual manera que el script planteado en el Arduino Yun se colocan los valores promedio para las variables y se agrega un componente aleatorio.

2.6 Visualización de la aplicación web

Los elementos desarrollados que conforman la aplicación web se encuentran listos y desplegados en sus respectivas plataformas (aplicación web alojada en Heroku y Base de datos alojada en Microsoft Azure). La conexión entre la base de datos y la aplicación web es exitosa. Se puede observar a través de un navegador la aplicación con los datos de prueba ingresados a través del script realizado en Python.

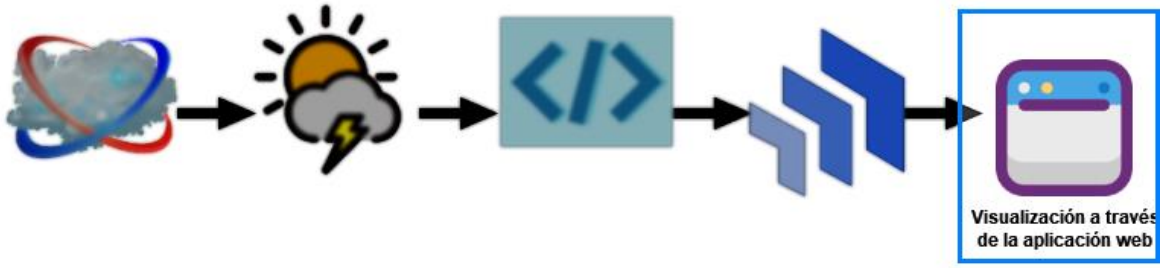


Figura 2.59 Etapa 5: Visualización de la aplicación web.

Para poder acceder a la aplicación web debemos ingresar la URL proporcionada por la plataforma que aloja la aplicación. La aplicación recibirá el nombre de invernadero-epn, que fue el nombre asignado a la aplicación creada en Heroku. Este nombre estará seguido por heroku.com. otra manera de acceder a la aplicación es a través del panel de control de Heroku. Se debe dar clic en el botón “Open app” para acceder a la aplicación.

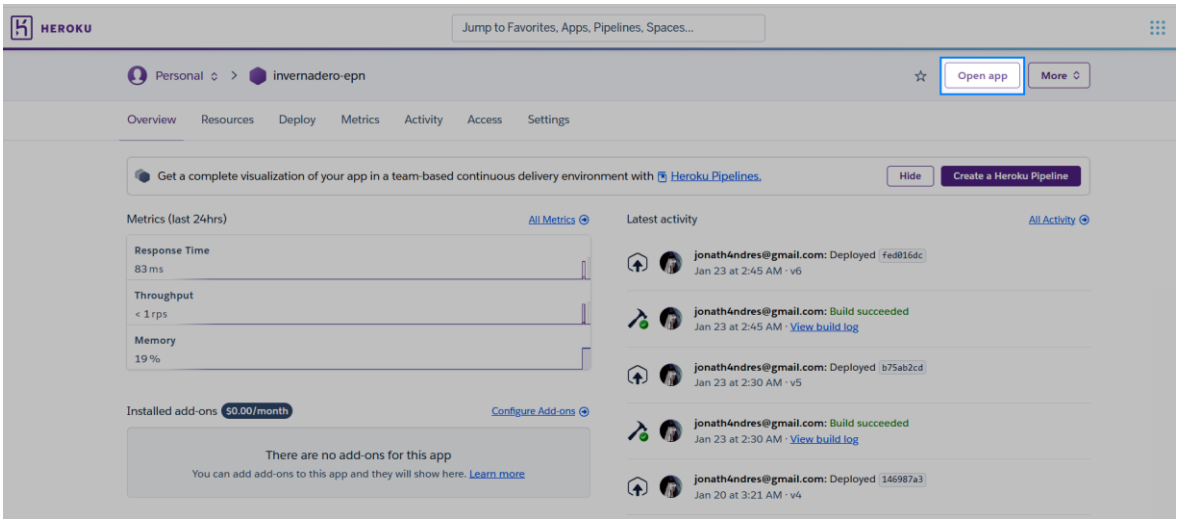


Figura 2.60 Botón para acceder a la aplicación desde el panel de control web de Heroku.

Al acceder podremos observar la aplicación web, algunos datos han sido enviados para poder observar su funcionamiento.

Figura 2.61 Aplicación web desplegada en la nube.

2.6.1 Conexión de Arduino Yun a internet

Como se vio anteriormente el dispositivo Arduino Yun tiene capacidades para conexión a internet mediante Ethernet y Wifi. También cuenta con un microcontrolador basado en Linux, esto le permite usar varios protocolos de internet y servicios web.

Al energizar el dispositivo se lo encuentra en las redes locales, esto se puede acceder a través de un dispositivo que tenga Wifi. El siguiente paso consiste en seleccionar la red correspondiente al Arduino Yun, se debe ingresar a la dirección IP del dispositivo que provee la documentación: 192.168.240.1 [42]. Una vez dentro se pedirá usuario/contraseña que para este caso son: root/dragino.

Al ingresar por primera vez se mostrará la información general del dispositivo, para realizar la configuración se debe ingresar a la opción “SYSTEM”. Es conveniente colocar el dispositivo en la zona horaria adecuada para que no existan confusiones. Dentro de esta ventana además se puede colocar otra contraseña, lo cual es recomendable por motivos de seguridad y cambiar el nombre de con el que se identifica el dispositivo de manera inalámbrica.

Al contar con Wifi detectará las redes locales, de estas seleccionar la red que proveerá de internet al dispositivo e ingresar sus credenciales. Luego de unos momentos la conexión será exitosa, para ingresar nuevamente al panel de configuración en lugar de la dirección IP por defecto se debe ingresar la dirección asignada al dispositivo dentro de la red.

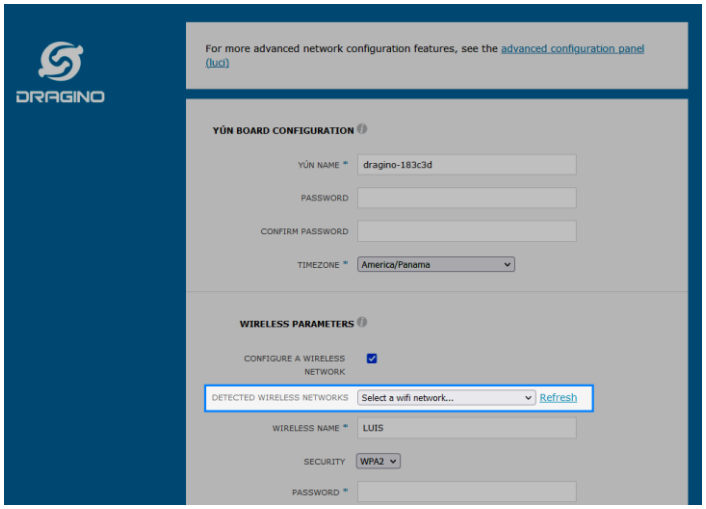


Figura 2.62 Selección de la red de área local que brinde acceso a internet al Arduino a través de Wifi

Cuando el dispositivo se encuentre dentro de nuestra red de área local es posible enviar sketches para que sean ejecutados. Dentro de Arduino IDE dirigirse a la pestaña “Tools” aquí se podrá instalar la librería para poder identificar el Shield.

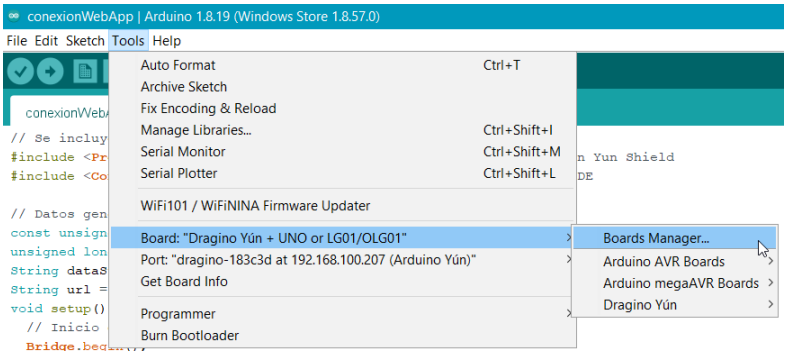


Figura 2.63 Instalación de las librerías que permiten la identificación del Arduino Yun

Al dar clic en la ventana de búsqueda ingresar la palabra “Dragino” e instalar la librería mostrada a continuación.

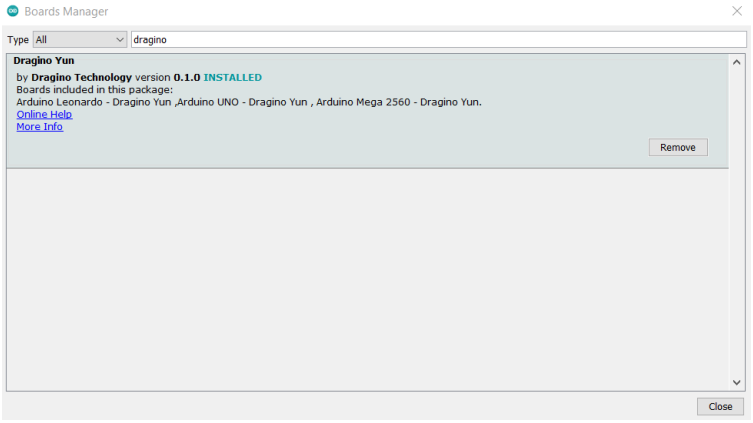


Figura 2.64 Librería Dragino Yun

Una vez instalada la librería en la pestaña Tools->Boards->Dragino Yun seleccionar la configuración usada que para el caso de este proyecto es Dragino Yun + UNO. Ya seleccionada aparecerá en parte inferior izquierda de la ventana de Arduino IDE la configuración y la dirección IP asignada al dispositivo dentro de la red de área local.

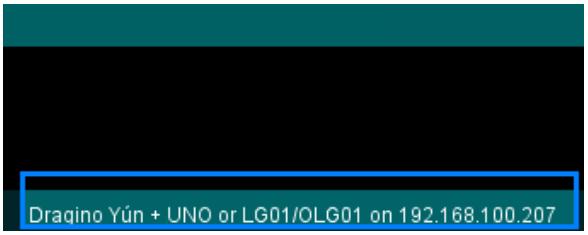


Figura 2.65 Información del dispositivo mostrada en Arduino IDE.

El funcionamiento de esta configuración Dragino Yun + UNO es la misma que la de un Arduino Yun debido a que cuentan con características muy similares. La programación para ambos dispositivos es la misma al igual que el uso de las librerías para el desarrollo del sketch.

Para poder hacer los comandos que permitan enviar la información recibida del invernadero a la aplicación web desplegada en la nube es necesario el uso de la librería '<Process.h>' es una parte del paquete "Bridge". Permite al usuario ejecutar comandos Linux, provee funciones para crear, correr e interactuar con proceso en Linux desde el dispositivo.

Otra librería usada es '<Process.h>', permite al usuario interactuar con la línea de comandos de Linux. Provee funciones para enviar comandos a la Shell de Linux, leer su respuesta y redirigir dicha respuesta a distintas localizaciones. En el proyecto sirve para poder observar desde el Plot Serial el comando que se enviará hacia la aplicación web en la nube.

El sketch construido recibe los datos del invernadero y crea un comando usando CURL para enviarlos hacia la aplicación web, para las pruebas se ha definido que se envíen cada minuto, pero efectivamente el dispositivo puede enviar la información a cualquier hora y la aplicación desplegada de igual manera puede recibir dicha información mientras este activa.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 RESULTADOS

Los resultados del proyecto fueron satisfactorios, la aplicación web se encuentra desplegada a en la dirección: https://invernadero-epn.herokuapp.com/. La base de datos se encuentra desplegada en la nube por razones de seguridad esta no se debe acceder directamente, el acceso lo realiza la aplicación web a través de la cadena de conexión. Finalmente, el Arduino Yun envía los datos exitosamente, el código fuente del sketch usado se encuentra en el ANEXO VII.

3.1.1 Aplicación Web

Mediante el navegador de preferencia al dirigirse a la dirección https://invernadero-epn.herokuapp.com/. Se presenta la gráfica junto con los últimos datos recibidos por el invernadero.

A continuación, se presenta la carátula del proyecto en donde se encuentra información del proyecto, su nombre, nombre del autor y director.

Figura 3.66 Carátula de la aplicación web.

Al dar clic en ingresar se mostrará la página principal de la aplicación web. En la parte superior izquierda encontramos los botones para dirigirse a la carátula o a la información histórica recibida por el invernadero. Después se muestra la fecha en la cual la última lectura fue recibida. Siguiente se encuentra unas tarjetas que nos indican los valores de las ultimas variables recibidas por el invernadero junto con una breve descripción de la

variable. En la parte inferior se muestra una gráfica con los valores recibidos en el último día, la gráfica se verá completa al bajar en la página mostrada.

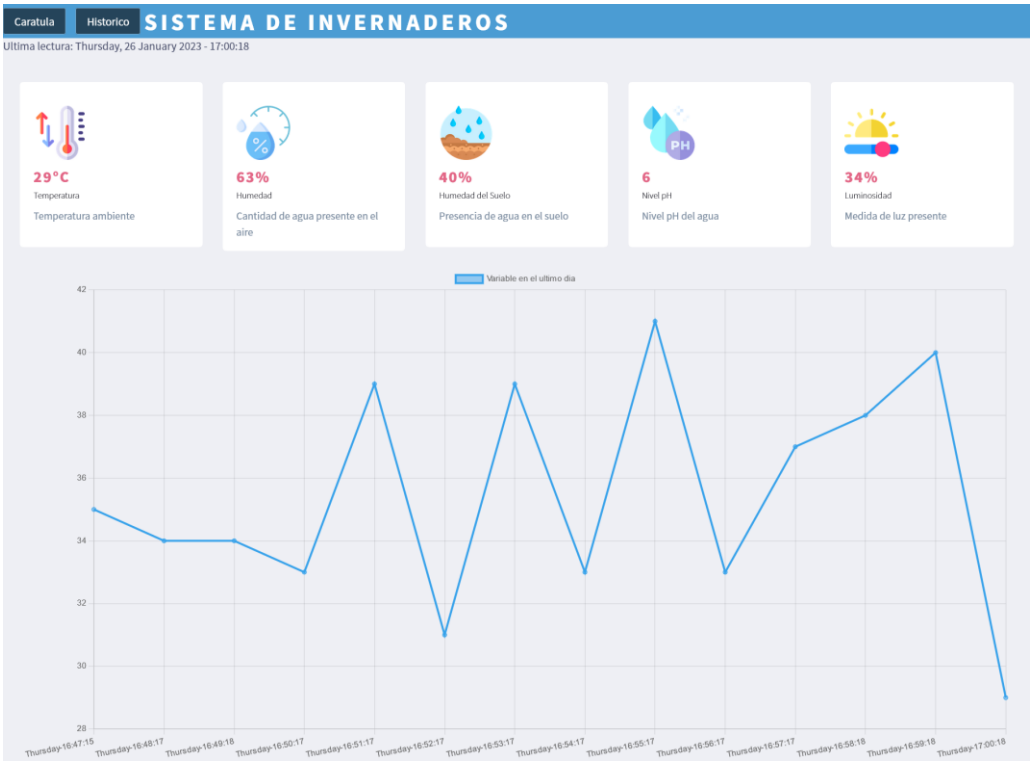


Figura 3.67 Captura de la página web principal mostrando la gráfica de la variable Temperatura.

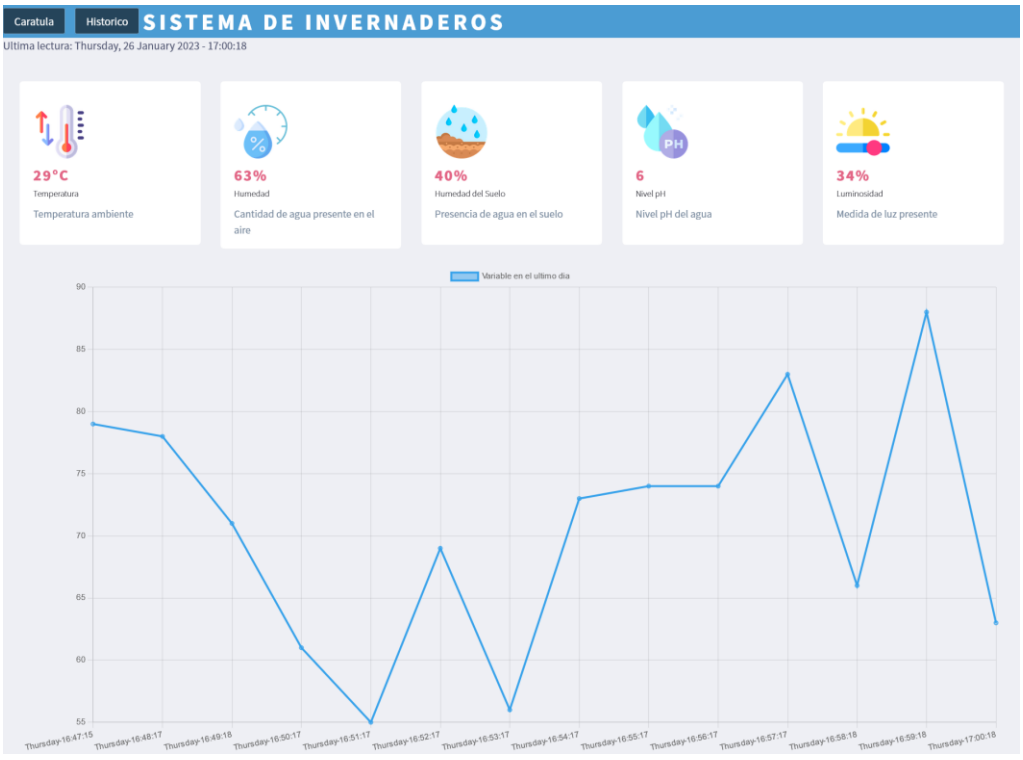


Figura 3.68 Captura de la página web principal mostrando la gráfica de la variable Humedad.

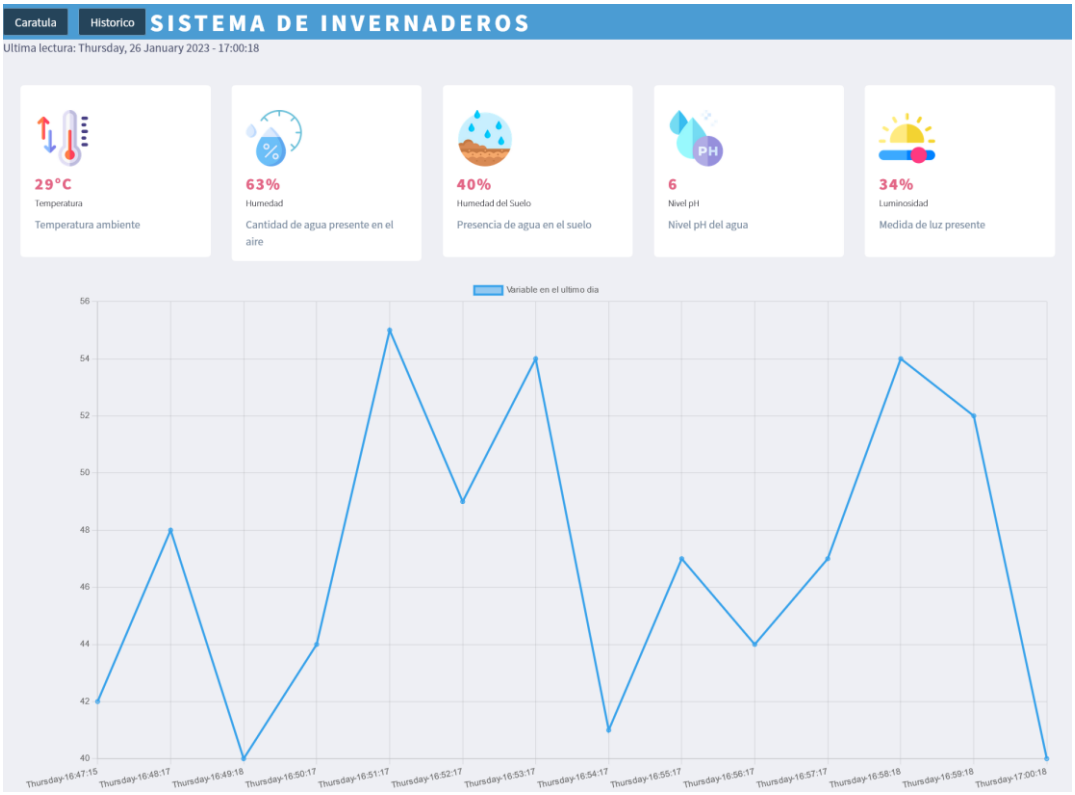


Figura 3.69 Captura de la página web principal mostrando la gráfica de la variable Humedad del Suelo.

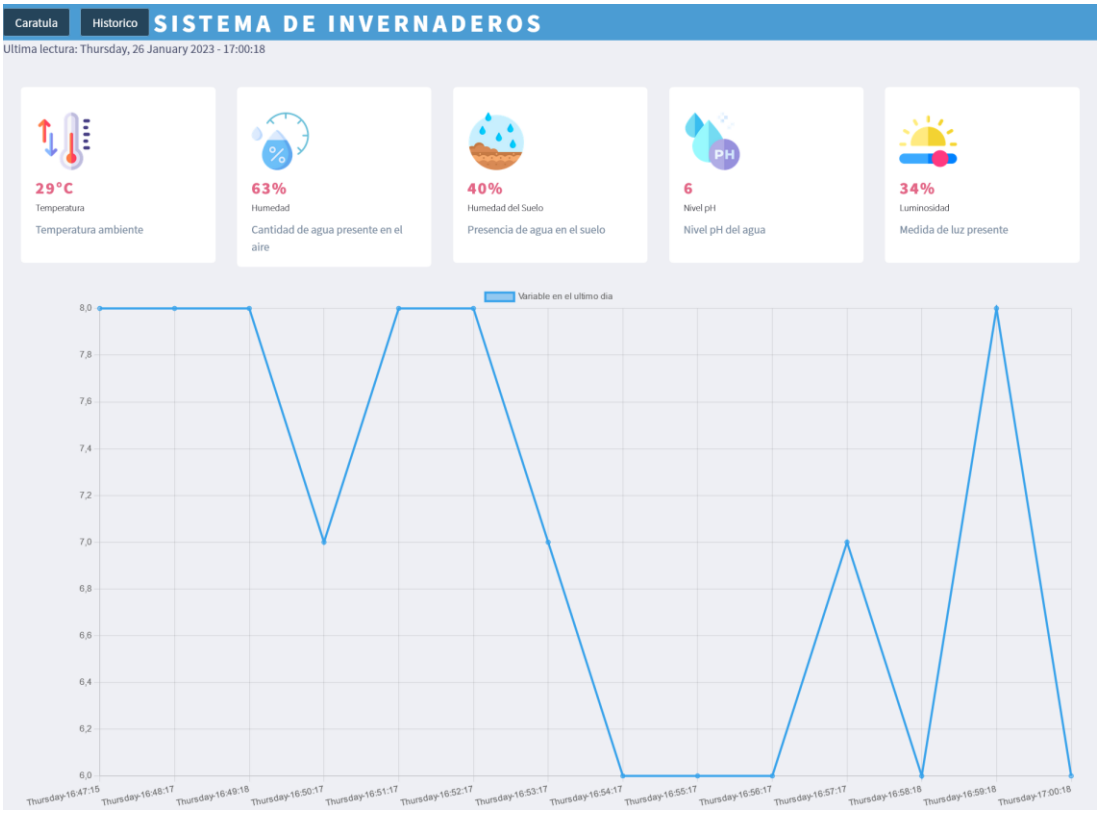


Figura 3.70 Captura de la página web principal mostrando la gráfica de la variable Nivel PH.

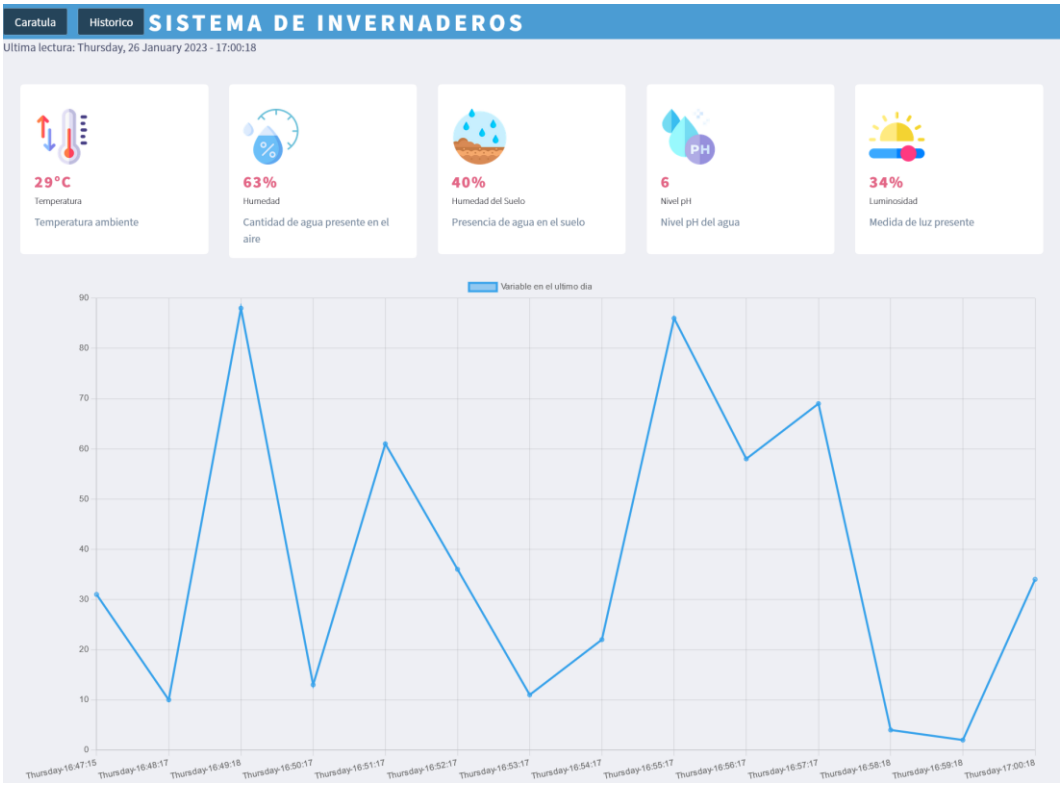


Figura 3.71 Captura de la página web principal mostrando la gráfica de la variable Luminosidad.

Para poder observar las gráficas de las demás variables se debe dar clic sobre la tarjeta correspondiente a la variable y la gráfica se actualizará de manera automática. No se realizará una recarga de la página mostrada.

Caratula Principal **SISTEMA DE INVERNADEROS**

Fecha UTC	Temperatura	Humedad	Humedad de suelo	Nivel de PH	Luminosidad
2023-01-20 08:40:16	31	69	56	8	3
2023-01-20 08:45:21	30	56	60	6	61
2023-01-20 08:45:28	34	73	54	8	97
2023-01-20 08:45:39	31	80	42	6	78
2023-01-20 08:47:00	28	79	40	6	29
2023-01-20 08:47:07	31	70	45	7	91
2023-01-26 16:47:15	35	79	42	8	31
2023-01-26 16:48:17	34	78	48	8	10
2023-01-26 16:49:18	34	71	40	8	88
2023-01-26 16:50:17	33	61	44	7	13
2023-01-26 16:51:17	39	55	55	8	61
2023-01-26 16:52:17	31	69	49	8	36
2023-01-26 16:53:17	39	56	54	7	11
2023-01-26 16:54:17	33	73	41	6	22
2023-01-26 16:55:17	41	74	47	6	86
2023-01-26 16:56:17	33	74	44	6	58
2023-01-26 16:57:17	37	83	47	7	69
2023-01-26 16:58:18	38	66	54	6	4
2023-01-26 16:59:18	40	88	52	8	2
2023-01-26 17:00:18	29	63	40	6	34

Figura 3.72 Página que muestra el histórico de las variables almacenadas.

Al dirigirse en la pestaña histórico se muestran todos los datos recibidos por parte del invernadero al igual que las fechas en las cuales fueron receptados por la aplicación web.

3.1.2 Base de datos

La información correspondiente a la base de datos se la puede encontrar al ingresar al portal de Azure. En la parte inferior del servicio se encuentran los gráficos del uso de la base de datos, de estos podemos ver que:

1. CPU y memoria: Al enviar o no enviar datos su uso se mantiene constante y por debajo del 50% esto nos indica que podríamos escalar hacia abajo o dejar el procesamiento restante.
2. Almacenamiento: Al ser datos muy simples usan muy poco espacio de disco duro.
3. Conexiones de BD: Las conexiones se mantienen constantes, la aplicación web realiza estas conexiones y no tienen mucha variación.
4. Rendimiento: Nos indica el éxito en la escritura, se observa que los datos en efecto llegan a la base de datos.

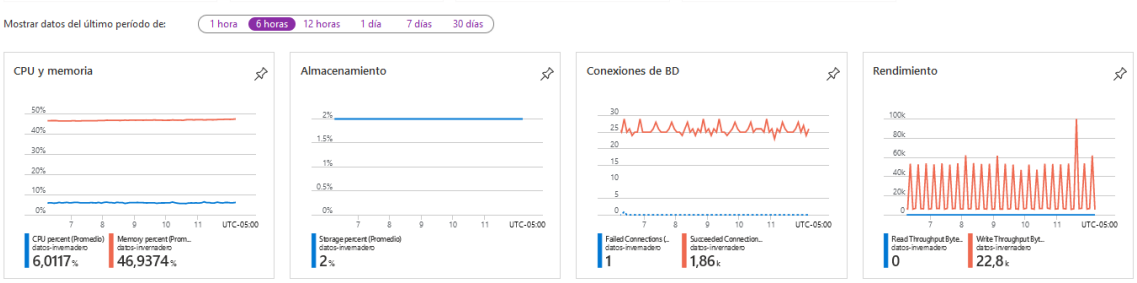


Figura 3.73. Gráficos generados del uso de la base de datos.

La administración de la base de datos se la hace a través del portal de Azure.

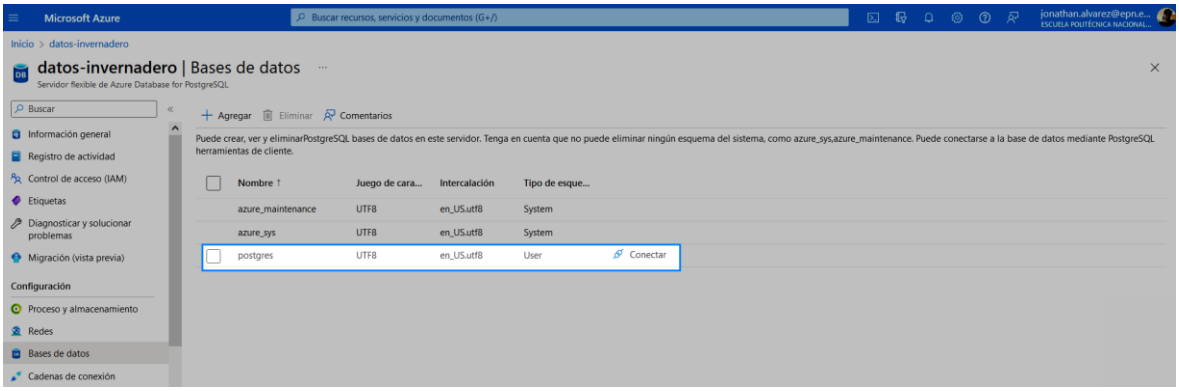


Figura 3.74 Conexión a través de la consola de Azure a la base de datos.

Dentro del portal se encuentra dentro de la opción Base de datos ubicada en el panel izquierdo la posibilidad de conectarse a la base datos. Para poder realizar la conexión es necesaria ingresar la contraseña de la base de datos. Una vez dentro se pueden ejecutar comandos de PostgresSQL, a continuación, se muestra el comando ejecutado para visualizar todos los valores almacenados en la tabla.

```
postgres=> SELECT * FROM invernadero;
 id |         content          |          fecha          | temperatura | humedad | humsuelo | nivelPH | luminosidad
----+--------------------------+-------------------------+-------------+---------+----------+---------+-------------
 28 | Localizacion invernadero | 2023-01-20 08:40:16.761752 |          31 |      69 |       56 |       8 |           3
 29 | Localizacion invernadero | 2023-01-20 08:45:21.760445 |          30 |      56 |       60 |       6 |          61
 30 | Localizacion invernadero | 2023-01-20 08:45:28.239135 |          34 |      73 |       54 |       8 |          97
 31 | Localizacion invernadero | 2023-01-20 08:45:39.038717 |          31 |      80 |       42 |       6 |          78
 32 | Localizacion invernadero | 2023-01-20 08:47:00.428635 |          28 |      79 |       40 |       6 |          29
 33 | Localizacion invernadero | 2023-01-20 08:47:07.661472 |          31 |      70 |       45 |       7 |          91
 34 | EnviadoArduino           | 2023-01-26 16:47:15.436142 |          35 |      79 |       42 |       8 |          31
 35 | EnviadoArduino           | 2023-01-26 16:48:17.658    |          34 |      78 |       48 |       8 |          10
 36 | EnviadoArduino           | 2023-01-26 16:49:18.335586 |          34 |      71 |       40 |       8 |          88
 37 | EnviadoArduino           | 2023-01-26 16:50:17.655969 |          33 |      61 |       44 |       7 |          13
 38 | EnviadoArduino           | 2023-01-26 16:51:17.69287  |          39 |      55 |       55 |       8 |          61
 39 | EnviadoArduino           | 2023-01-26 16:52:17.740943 |          31 |      69 |       49 |       8 |          36
 40 | EnviadoArduino           | 2023-01-26 16:53:17.927837 |          39 |      56 |       54 |       7 |          11
 41 | EnviadoArduino           | 2023-01-26 16:54:17.901648 |          33 |      73 |       41 |       6 |          22
 42 | EnviadoArduino           | 2023-01-26 16:55:17.87413  |          41 |      74 |       47 |       6 |          86
 43 | EnviadoArduino           | 2023-01-26 16:56:17.976553 |          33 |      74 |       44 |       6 |          58
 44 | EnviadoArduino           | 2023-01-26 16:57:17.938103 |          37 |      83 |       47 |       7 |          69
 45 | EnviadoArduino           | 2023-01-26 16:58:18.005993 |          38 |      66 |       54 |       6 |           4
 46 | EnviadoArduino           | 2023-01-26 16:59:18.050362 |          40 |      88 |       52 |       8 |           2
 47 | EnviadoArduino           | 2023-01-26 17:00:18.105452 |          29 |      63 |       40 |       6 |          34
(20 rows)
```

Figura 3.75 Valores de la tabla mostrados a través de la consola de Azure.

La consola de administración de Azure brinda la posibilidad de administrar la base de datos cuando contemos con una conexión a internet. Como se mencionó anteriormente el uso de la base de datos implica un costo que será descontado de los créditos proporcionados a la cuenta estudiantil. Un resumen del costo de la base de datos se muestra a continuación.

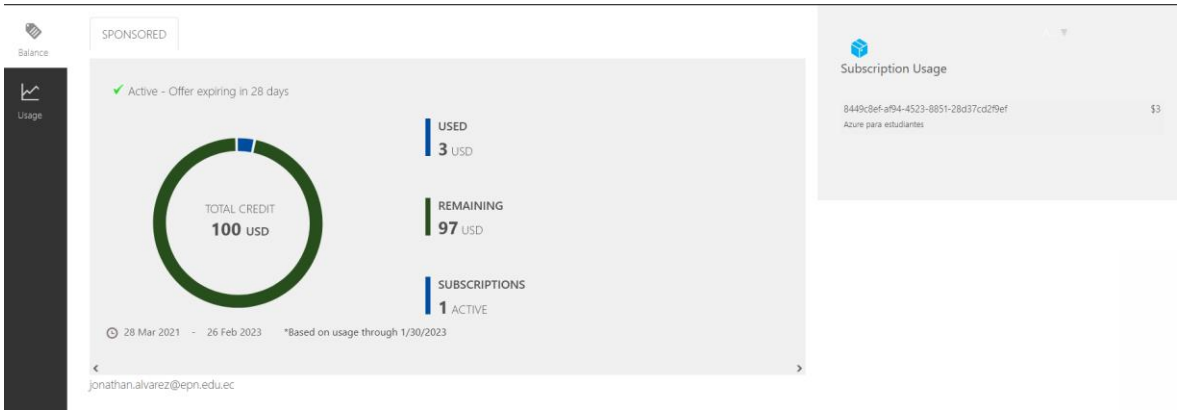


Figura 3.76 Información del crédito disponible en Azure.

El costo no ha sido considerable debido a que al momento de seleccionar las características de procesamiento se tuvo en cuenta escoger la opción mínima. Debido a

que Azure permite escalar los servicios no es un problema aumentar la capacidad en un futuro en caso de que los recursos elegidos hayan sido insuficientes.

3.1.3 Envío de datos a través de Arduino Yun

Él envió de datos desde el Arduino Yun se lo puede observar a través del Serial Plot en Arduino IDE. Para comprobar que los valores de las variables llegan a la base de datos seleccionamos uno del grupo de datos enviados y se observa la cadena que será enviada hacia la aplicación web:

EnviadoArduino&temperatura=34&humedad=71&humsuelo=40&nivelPH=8&luminosidad=88

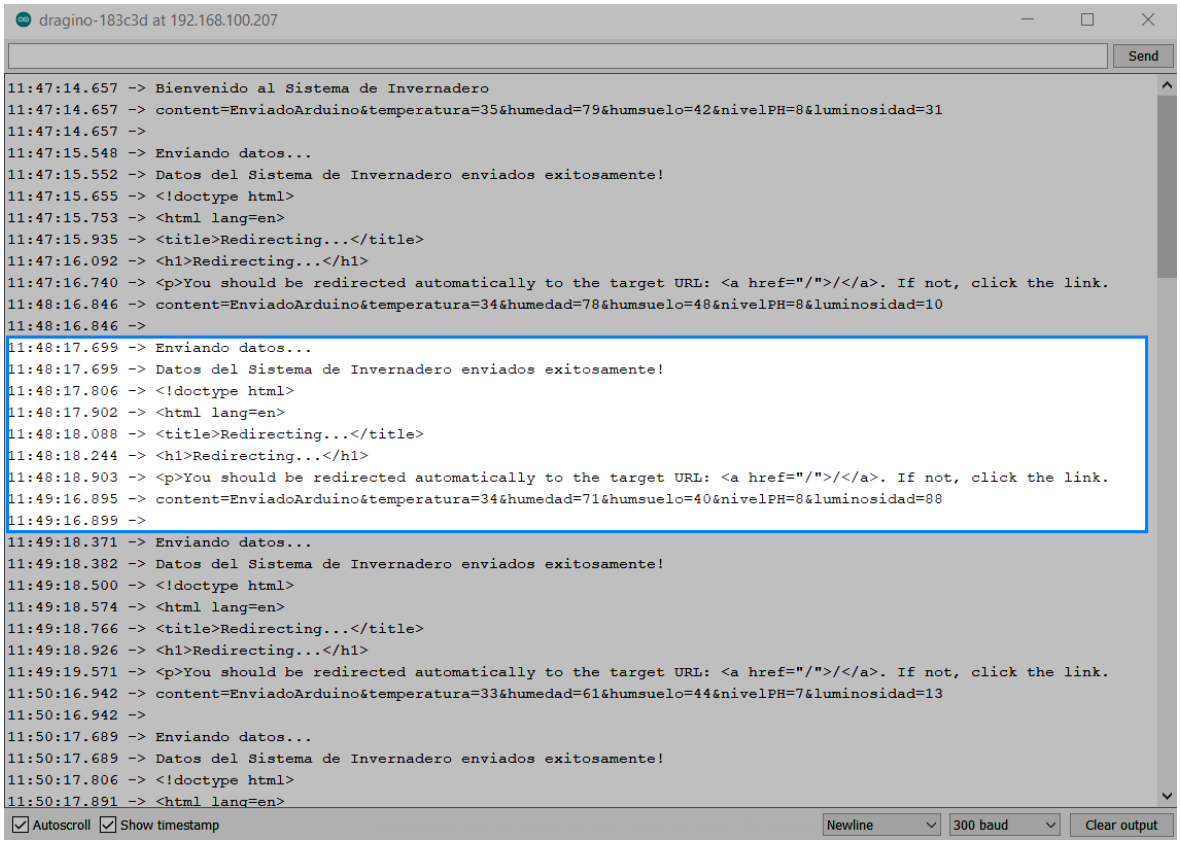


Figura 3.77 Arduino Yun enviado datos hacia la aplicación web.

Los valores mostrados se pueden encontrar en la aplicación web.

Fecha UTC	Temperatura	Humedad	Humedad de suelo	Nivel de PH	Luminosidad
2023-01-20 08:40:16	31	69	56	8	3
2023-01-20 08:45:21	30	56	60	6	61
2023-01-20 08:45:28	34	73	54	8	97
2023-01-20 08:45:39	31	80	42	6	78
2023-01-20 08:47:00	28	79	40	6	29
2023-01-20 08:47:07	31	70	45	7	91
2023-01-26 16:47:15	35	79	42	8	31
2023-01-26 16:48:17	34	78	48	8	10
2023-01-26 16:49:18	34	71	40	8	88
2023-01-26 16:50:17	33	61	44	7	13
2023-01-26 16:51:17	39	55	55	8	61
2023-01-26 16:52:17	31	69	49	8	36
2023-01-26 16:53:17	39	56	54	7	11
2023-01-26 16:54:17	33	73	41	6	22
2023-01-26 16:55:17	41	74	47	6	86
2023-01-26 16:56:17	33	74	44	6	58

Figura 3.78 Valores enviados desde el Arduino Yun mostrados desde la aplicación web.

3.2 Conclusiones

La solución propuesta para presentar los valores obtenidos del invernadero enviados a través de un Arduino Yun conectado a internet que se almacenan en una base de datos y son presentados al usuario mediante una aplicación web ha sido implementada y probada de manera satisfactoria. Los resultados muestran que la solución además de ser efectiva es eficiente, pero además es amigable para el usuario y accesible. Les ofrece a los agricultores una herramienta conveniente para el monitoreo que lleva a la optimización de las condiciones del cultivo en los invernaderos.

El uso de un Arduino Yun para recibir los datos del invernadero y transmitir los mismos a través de internet asegura que se tendrá poco consumo de energía además de transmisión confiable de los datos, mientras que la base de datos y la aplicación web al usar las gráficas para cada una de las variables permite análisis de los datos y su presentación en tiempo real.

La solución presentada tiene múltiples ventajas como el ser económica, aunque no fue objetivo propuesto para la elaboración del trabajo lograr el desarrollo sin costo se trató de considerar alternativas gratuitas y ofertas estudiantiles para acceder a los recursos necesarios como fue el almacenamiento de datos y el alojamiento de la aplicación en la nube. Otra ventaja que presento la solución es ser escalable, en los resultados se pudo observar que el costo por la parte de la base de datos fue muy bajo en el periodo de tiempo que se desarrolló el proyecto, si fuese necesario tener mas recursos basta con subir a un nivel mayor de servicio para obtener mayor almacenamiento y procesamiento. Además, la solución es accesible desde cualquier lugar que tenga acceso a internet, permitiendo a la persona que realiza el monitoreo acceder a los datos desde amplias distancias o desde sus hogares, esto es conveniente y flexible.

Para el desarrollo del trabajo fue necesario investigar diferentes alternativas de servicios en la nube, al considerar sus ventajas y desventajas entre estas que sean alternativas sin costo o pagadas. Se debe considerar que al seleccionar la opción más económica fue necesario pasar más tiempo investigando, probando errores y configurando los sistemas. Por otro lado, pagar por un proveedor de servicios en la nube puede ahorrar tiempo.

Las propuestas de desarrollo de aplicaciones y servicios en la nube ofrecidos a estudiantes con cuentas institucionales son una gran manera para que los estudiantes conozcan las nuevas tecnologías y productos que ofrecen las empresas. Además, como se observó en la elaboración del proyecto permiten efectivamente desarrollar una aplicación con las ofertas brindadas.

Existen algunas limitaciones que deben ser consideradas como la necesidad de medidas mas robustas de seguridad, los datos del invernadero son sensibles y deben ser protegidos contra acceso no autorizado o brechas de seguridad. Esto requerirá mejoras en el manejo de seguridad especialmente en la parte de la conexión entre el Arduino Yun y la base datos para asegurar integridad de los datos capturados y almacenados.

3.3 Recomendaciones

Analizar los posibles problemas que podrían presentarse con el uso de la interfaz de usuario por parte de las personas que realizan el monitoreo. A pesar de que se trató de hacer la interfaz amigable para el usuario pueden existir áreas en la presentación que se pueda mejorar.

Explorar la integración de otro tipo de dispositivos para el envío de datos y con otras soluciones existentes de administración de invernaderos para incrementar la practicidad y eficiencia de la solución propuesta.

Investigar las posibilidades que brinda la cuenta institucional para el desarrollo de aplicaciones y otros beneficios. Como se vio en el proyecto GitHub Student Pack ofrece gran número de servicios tanto para el despliegue y creación de aplicaciones web como para otros proyectos.

Considerar realizar estudios adicionales relacionados al escalamiento de la solución propuesta. Si se trabaja con muchas variables enviadas en periodos muy cortos de tiempo el uso de recursos será mayor que el presentado en este trabajo, de ser ese el caso es recomendable estudiar dentro de un periodo de tiempo cuanto espacio será requerido y proyectar dichos resultados para obtener la cantidad de recursos necesarios a largo plazo.

Mejorar las medidas de seguridad con respecto a la base de datos, aplicación web y del Arduino Yun para asegurar la integridad de los datos guardados y que las variables provenientes del invernadero no sean manipuladas.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] J. E. S. Anthony, «Auto GMS: An Automated Greenhouse Monitoring System of Abiotic Factors for Leafy Vegetables Production», *KnE Soc. Sci.*, pp. 682-707, jun. 2018, doi: 10.18502/kss.v3i6.2413.

[2] «A Proposal of Greenhouse Control Using Wireless Sensor Networks». https://doi.org/10.13031/2013.21878 (accedido 23 de noviembre de 2022).

[3] V. C. Patil, K. A. Al-Gaadi, D. P. Biradar, y M. Rangaswamy, «INTERNET OF THINGS (IOT) AND CLOUD COMPUTING FOR AGRICULTURE: AN OVERVIEW», *Internet Things*, p. 6.

[4] U. Mc Carthy, I. Uysal, R. Badia-Melis, S. Mercier, C. O'Donnell, y A. Ktenioudaki, «Global food security – Issues, challenges and technological solutions», *Trends Food Sci. Technol.*, vol. 77, pp. 11-20, jul. 2018, doi: 10.1016/j.tifs.2018.05.002.

[5] C. Hairu, M. Hanafi, S. M. Shafie, W. F. Fazlil Ilahi, y S. Mashohor, «Remote Monitoring of Surface Temperature and Moisture Content in Urban Greenhouse», en *2020 IEEE 8th Conference on Systems, Process and Control (ICSPC)*, dic. 2020, pp. 101-105. doi: 10.1109/ICSPC50992.2020.9305786.

[6] P. K. Tripathy, A. K. Tripathy, A. Agarwal, y S. P. Mohanty, «MyGreen: An IoT-Enabled Smart Greenhouse for Sustainable Agriculture», *IEEE Consum. Electron. Mag.*, vol. 10, n.º 4, pp. 57-62, jul. 2021, doi: 10.1109/MCE.2021.3055930.

[7] A. Sagheer, M. Mohammed, K. Riad, y M. Alhajhoj, «A Cloud-Based IoT Platform for Precision Control of Soilless Greenhouse Cultivation», *Sensors*, vol. 21, n.º 1, Art. n.º 1, ene. 2021, doi: 10.3390/s21010223.

[8] United States: Commerce Department: National Institute of Standards and Technology (NIST), P. M. Mell, P. M. Mell, P. M. Mell, T. Grance, y National Institute of Standards and Technology (U.S.): Computer Security Division., «The NIST Definition of Cloud Computing». Commerce Department, Gaithersburg, MD, 1 de enero de 2011. Accedido: 1 de diciembre de 2022. [En línea]. Disponible en: http://dx.doi.org/10.6028/NIST.SP.800-145

[9] «Arduino Yún | Arduino Documentation». https://docs.arduino.cc/retired/boards/arduino-yun (accedido 8 de diciembre de 2022).

[10] «Arduino Yun Shield projects», *Hackster.io*. https://create.arduino.cc/projecthub/products/arduino-yun-shield (accedido 8 de diciembre de 2022).

[11] Y. Shield, «Linux, WiFi shield for Arduino», p. 1.

[12] S. Mathew, «Overview of Amazon Web Services», 2014.

[13] «What is Microsoft Azure and How Does It Work [Updated] | Simplilearn», *Simplilearn.com*. https://www.simplilearn.com/tutorials/azure-tutorial/what-is-azure (accedido 31 de diciembre de 2022).

[14] «How Heroku Works | Heroku Dev Center». https://devcenter.heroku.com/articles/how-heroku-works (accedido 3 de enero de 2023).

[15] PythonAnywere, «The PythonAnywhere help pages», *PythonAnywhere help*, 13 de mayo de 2015. https://help.pythonanywhere.com/pages/ (accedido 5 de enero de 2023).

[16] «What is a Virtual Machine? | VMware Glossary». https://www.vmware.com/topics/glossary/content/virtual-machine.html (accedido 5 de enero de 2023).

[17] E. P. Nacional, «Correo Institucional», *Escuela Politécnica Nacional*, 1 de octubre de 2014. https://www.epn.edu.ec/correo-institucional/ (accedido 23 de enero de 2023).

[18] tfosmark, «About the Azure for Students program (Azure Education Hub)». https://learn.microsoft.com/en-us/azure/education-hub/azure-dev-tools-teaching/azure-students-program (accedido 10 de enero de 2023).

[19] «Azure para estudiantes universitarios: detalles de la oferta | Microsoft Azure». https://azure.microsoft.com/es-es/offers/ms-azr-0170p/ (accedido 10 de enero de 2023).
[20] S. Hamidah, Y. F. Arifin, y A. Fitriani, «MICRO CLIMATE ASSESSMENT OF MEDICINAL PLANT HABITAT FOR THE FIRST STEP OF DOMESTICATION», vol. 9, 2018.
[21] «Soil pH and Nutrient Availability». https://www.horiba.com/int/water-quality/applications/agriculture-crop-science/soil-ph-and-nutrient-availability/ (accedido 18 de enero de 2023).
[22] Y. Zheng *et al.*, «Suitable soil moisture contents for water use efficiency and saponins accumulation in Panax notoginseng», *Chin. Herb. Med.*, vol. 13, n.º 2, pp. 267-273, abr. 2021, doi: 10.1016/j.chmed.2020.10.002.
[23] J. S. U. 23 de febrero de 2021 | P. 4 de abril de 2019, «What is cURL and how does it relate to APIs?», *IBM Developer*. https://developer.ibm.com/articles/what-is-curl-command/ (accedido 18 de enero de 2023).
[24] «POST - HTTP | MDN». https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST (accedido 18 de enero de 2023).
[25] «What is Python? Executive Summary», *Python.org*. https://www.python.org/doc/essays/blurb/ (accedido 11 de enero de 2023).
[26] «Applications for Python», *Python.org*. https://www.python.org/about/apps/ (accedido 11 de enero de 2023).
[27] «WebFrameworks - Python Wiki». https://wiki.python.org/moin/WebFrameworks (accedido 11 de enero de 2023).
[28] «Welcome to Flask — Flask Documentation (2.2.x)». https://flask.palletsprojects.com/en/2.2.x/ (accedido 11 de enero de 2023).
[29] «Documentation for Visual Studio Code». https://code.visualstudio.com/docs (accedido 11 de enero de 2023).
[30] «Quickstart — Flask Documentation (2.2.x)». https://flask.palletsprojects.com/en/2.2.x/quickstart/ (accedido 11 de enero de 2023).
[31] «Using a virtual environment with your flask app». https://pythonhow.com/python-tutorial/flask/Using-a-virtual-environment-with-your-flask-app/ (accedido 11 de enero de 2023).
[32] «What is an ORM – The Meaning of Object Relational Mapping Database Tools», *freeCodeCamp.org*, 21 de octubre de 2022. https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/ (accedido 11 de enero de 2023).
[33] «Flask Templates — Python Beginners documentation». https://python-adv-web-apps.readthedocs.io/en/latest/flask3.html (accedido 11 de enero de 2023).
[34] «Etiquetas HTML. HTML. Páginas web HTML y hojas de estilo CSS. Bartolomé Sintes Marco. www.mclibre.org». https://www.mclibre.org/consultar/htmlcss/html/html-etiquetas.html (accedido 11 de enero de 2023).
[35] A. L. Cruz, «Tabla resumen de propiedades CSS y sus valores», *Eniun*, 12 de octubre de 2019. https://www.eniun.com/resumen-tabla-propiedades-css-valores/ (accedido 11 de enero de 2023).
[36] «What is JavaScript? - Learn web development | MDN». https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript (accedido 24 de enero de 2023).
[37] «Chart.js | Chart.js». https://www.chartjs.org/docs/latest/ (accedido 24 de enero de 2023).
[38] «About - Git». https://git-scm.com/about/branching-and-merging (accedido 24 de enero de 2023).
[39] «What Is Gunicorn, and What Does It Do?», *vsupalov.com*, 23 de abril de 2019. https://vsupalov.com/what-is-gunicorn/ (accedido 24 de enero de 2023).
[40] «PostgreSQL: About». https://www.postgresql.org/about/ (accedido 24 de enero de 2023).

[41] E. Erkec, «SQL Connection Strings tips», *SQL Shack - articles about database auditing, server performance, data recovery, and more*, 24 de septiembre de 2021. https://www.sqlshack.com/sql-connection-strings-tips/ (accedido 25 de enero de 2023).
[42] «YUN_SHIELD_USER_MANUAL_v1.0.pdf». Accedido: 26 de enero de 2023. [En línea]. Disponible en: https://www.dragino.com/downloads/downloads/YunShield/YUN_SHIELD_USER_MANUAL_v1.0.pdf

ANEXO I. Código fuente aplicación en Flask

```
from flask import Flask, render_template, url_for, request, redirect,
jsonify
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime, timedelta

app = Flask(__name__)
# String de conexion usando el  URI DB Azure postgresql
app.config['SQLALCHEMY_DATABASE_URI']                                    =
'postgresql://jon4lvarez:{contrasena}@datos-
invernadero.postgres.database.azure.com/postgres'
# String de conexion para Railwayapp, brinda uso por 20 dias facil de
usar
#app.config['SQLALCHEMY_DATABASE_URI']                                   =
'postgresql://postgres:{contrasena}@containers-us-west-
72.railway.app:6130/railway'
# Usado para pruebas locales
# app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
# Crea la instancia de la clase de la base de datos
db = SQLAlchemy(app)

# Crea las columnas de la base de datos, con los parametros provenientes
del invernadero
class Invernadero(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    content = db.Column(db.String(200), nullable=False)
    fecha = db.Column(db.DateTime, default=datetime.utcnow)
    temperatura = db.Column(db.Integer)
    humedad = db.Column(db.Integer)
    humsuelo = db.Column(db.Integer)
    nivelPH = db.Column(db.Integer)
    luminosidad = db.Column(db.Integer)
    def __repr__(self):
        return '<Lectura %r>' % self.id

# Ruta principal de la aplicacion, si detecta una solicitud POST
# procedera a registrar los datos, si regestra un metodo GET
# mostrara los ultimos datos y el grafico de los datos de la variable
```

```
@app.route('/', methods=['POST', 'GET'])
def index():
    if request.method == 'POST':
        lect_content = request.form['content']
        lect_temperatura = request.form['temperatura']
        lect_humedad = request.form['humedad']
        lect_humsuelo = request.form['humsuelo']
        lect_nivelPH = request.form['nivelPH']
        lect_luminosidad = request.form['luminosidad']
        new_lectura=
Invernadero(content=lect_content,temperatura=lect_temperatura,humedad=l
ect_humedad,humsuelo=lect_humsuelo,nivelPH=lect_nivelPH,luminosidad=lec
t_luminosidad)
        try:
            db.session.add(new_lectura)
            db.session.commit()
            return redirect('/')
        except:
            return 'There was an issue adding your Invernadero'

    else:
        # Llamamos a la funcion para obtener variables relacionadas a
las mismas
        fecha_consulta, fecha_ult, ult = ultimas_fechas()
        # Consulta las lecturas presentadas en el ultimo dia para la
presentacion
        lecturas                                                    =
Invernadero.query.filter(Invernadero.fecha.between(fecha_consulta,
fecha_ult)).all()
        # Obtiene todas las lecturas de la base de datos
        lectura = Invernadero.query.order_by(Invernadero.fecha).all()
        fech, var, ides = [], [], []
        for lect in lecturas:
            fech.append(lect.fecha.strftime("%A-%H:%M:%S"))
            var.append(lect.temperatura)
            ides.append(lect.id)
        # Formato de la fecha de ultima lectura
        fecha_ult_cons=fecha_ult.strftime("%A, %d %B %Y - %H:%M:%S")
        return      render_template('index.html',      lectura=lectura,
fecha_ult_cons=fecha_ult_cons, fech=fech, var=var, ult=ult)
```

```
# Esta funcion permite modificar el grafico al dar clic a la variable
que se desea
# dependiendo del id de la tarjeta que contiene cada variable llama a la
variabe especifica
# y la funcion devuelve dichos valores
@app.route('/update-chart/<card_id>', methods=['GET'])
def update_chart(card_id):
    fecha_consulta, fecha_ult, ult = ultimas_fechas()
    lecturas                                                              =
Invernadero.query.filter(Invernadero.fecha.between(fecha_consulta,
fecha_ult)).all()
    card_id = int(card_id)
    new_data = []
    fech = []
    [fech.append(lect.fecha.strftime("%A-%H:%M:%S"))   for   lect   in
lecturas if 1 == 1]
    [new_data.append(lect.temperatura) for lect in lecturas if card_id
== 1]
    [new_data.append(lect.humedad) for lect in lecturas if card_id == 2]
    [new_data.append(lect.humsuelo) for lect in lecturas if card_id ==
3]
    [new_data.append(lect.nivelPH) for lect in lecturas if card_id == 4]
    [new_data.append(lect.luminosidad) for lect in lecturas if card_id
== 5]
    print(new_data)
    print(fech)
    var = new_data
    fecha_ult_cons=fecha_ult.strftime("%A, %d %B %Y - %H:%M:%S")
    return jsonify(new_data)

# La funcion devuelve las fechas para el proceso de los datos, incluye
la fecha de la
# ultima variable recibida, un tiempo de consulta definido de 24 horas
def ultimas_fechas():
    # Pedir la ultima entrada en la db
    ult = Invernadero.query.order_by(Invernadero.id.desc()).first()
    # La hora se maneja en UTC
    # Se toma las lecturas del ultimo dia
    fecha_ult = ult.fecha
```

```
    fecha_consulta = fecha_ult - timedelta(days=1,hours=0, minutes=0)
    return fecha_consulta, fecha_ult, ult

# Pagina de la caratula para la presentacion
@app.route('/caratula')
def caratula():
    title = "Caratula"
    return render_template('caratula.html', tittle=title)

# Pagina de los datos historicos de la aplicacion permite ver todos los
datos en una tabla
@app.route('/historico')
def historico():
    title = "Historico"
    lectura = Invernadero.query.order_by(Invernadero.fecha).all()
    return render_template('historico.html', lectura=lectura)

# Permite borrar datos en la tabla dado su ID
@app.route('/delete/<int:id>')
def delete(id):
    task_to_delete = Invernadero.query.get_or_404(id)
    try:
        db.session.delete(task_to_delete)
        db.session.commit()
        return redirect('/')
    except:
        return 'Existio un problema borrando el valor'

# Usado para inicializar la base de datos si esta vacia
@app.before_first_request
def create_tables():
    db.create_all()

if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0")
# IP de donde quiero aceptar las peticiones
```

ANEXO II. Código fuente del index.html

```
{% extends 'base.html' %}
{% block head %}
<title>Sistema de Invernaderos</title>
{% endblock %}
{% block body %}
<div class="wrapper">
      <div class="header">
            <div class="button-container">
                  <a href="/caratula"><button>Caratula</button></a>
                  <a href="/historico"><button>Historico</button></a>
            </div>
            <h2>Sistema de Invernaderos</h2>
      </div>
      <div   class="last_update">Ultima   lectura:   {{   fecha_ult_cons
}}</div>
      <div class="cards_wrap">
            <div class="card_item" id="card1" onclick="updateChart(1)">
                  <div class="card_inner">
                        <img src="/static/img/temp.png">
                        <div     class="variable">{{     ult.temperatura
}}°C</div>
                        <div class="nombre_variable">Temperatura</div>
                        <div             class="descripcion">Temperatura
ambiente</div>
                  </div>
            </div>
            <div class="card_item" id="card2" onclick="updateChart(2)">
                  <div class="card_inner">
                        <img src="/static/img/hum.png">
                        <div class="variable">{{ ult.humedad }}%</div>
                        <div class="nombre_variable">Humedad</div>
                        <div   class="descripcion">Cantidad   de   agua
presente en el aire</div>
                  </div>
            </div>

            <div class="card_item" id="card3" onclick="updateChart(3)">
                  <div class="card_inner">
```

```
                        <img src="/static/img/soil.png">
                        <div class="variable">{{ ult.humsuelo }}%</div>
                        <div     class="nombre_variable">Humedad     del
Suelo</div>
                        <div class="descripcion">Presencia de agua en el
suelo</div>
                  </div>
            </div>
            <div class="card_item" id="card4" onclick="updateChart(4)">
                  <div class="card_inner">
                        <img src="/static/img/ph.png">
                        <div class="variable">{{ ult.nivelPH }}</div>
                        <div class="nombre_variable">Nivel pH</div>
                        <div class="descripcion">Nivel pH del agua</div>
                  </div>
            </div>
            <div class="card_item" id="card5" onclick="updateChart(5)">
                  <div class="card_inner">
                        <img src="/static/img/lum.png">
                        <div     class="variable">{{     ult.luminosidad
}}%</div>
                        <div class="nombre_variable">Luminosidad</div>
                        <div    class="descripcion">Medida    de    luz
presente</div>
                  </div>
            </div>
      </div>
</div>
<!-- Generacion de la figura a partir de los datos provistos por el
invernadero -->
<div class="chart-container">
      <canvas id="miTabla"></canvas>
</div>
<script>
      var ctx = document.getElementById("miTabla").getContext("2d")
      var lineChart = new Chart(ctx, {
            type: "line",
            data: {
                  labels: {{ fech | tojson }},
      datasets: [
```

```
            {
                  label: "Variable en el ultimo dia ",
                  data: {{ var | tojson }}
                              }
      ]
                  },
      options: { }
            })

</script>

<!-- Secuencia de actualizacion de los datos al dar clic en las tarjetas
con la variables -->
<script>
      function updateChart(cardId) {
            axios.get('/update-chart/' + cardId)
            .then(response => {
                  console.log(response.data);
                  lineChart.config.data.datasets[0].data                =
response.data;
                  lineChart.update();
            })
            .catch(error => {
                  console.log(error);
            });
      }
</script>
{% endblock %}
```

ANEXO III. Código fuente historico.html

```
{% extends 'base.html' %}
{% block head %}
<title>Datos Historicos</title>
{% endblock %}
{% block body %}
<div class="wrapper">
    <div class="header">
        <div class="button-container">
            <a href="/caratula"><button>Caratula</button></a>
            <a href="/"><button>Principal</button></a>
        </div>
        <h2>Sistema de Invernaderos</h2>
    </div>
</div>
<div class="content">
    {% if lectura|length < 1 %} <h4>No existen datos ingresados!</h4>
        {% else %}
        <table class="table table-striped">
            <tr>th>Fecha UTC</th>
                <th>Temperatura</th>
                <th>Humedad</th>
                <th>Humedad de suelo</th>
                <th>Nivel de PH</th>
                <th>Luminosidad</th>tr>
            {% for lect in lectura %}
            <tr>
                <td>{{ lect.fecha.strftime('%Y-%m-%d %H:%M:%S') }}</td>
                <td>{{ lect.temperatura }}</td>
                <td>{{ lect.humedad }}</td>
                <td>{{ lect.humsuelo }}</td>
                <td>{{ lect.nivelPH }}</td>
                <td>{{ lect.luminosidad }}</td>
                <!-- <td>{{ lect.id }}</td>tr>
            {% endfor %}
        </table>
        {% endif %}
</div>
{% endblock %}
```

ANEXO IV. Código fuente caratula.html

```
{% extends 'base.html' %}
{% block head %}
<title>EPN</title>
<style>
    body {
      background-image:        url('/static/img/pexels-johannes-plenio-
1423601.jpg');
      background-size: cover;
      background-repeat: no-repeat;
    }
  </style>
{% endblock %}
{% block body %}
<div style="display: flex; justify-content: center; align-items: center;
height: 100vh;">
    <div style="width: 55%; text-align: center;">
      <!-- Your text content goes here -->
      <h1> Escuela Politécnica Nacional</h1>
      <h1>Facultad de Ingeniería Eléctrica y Electrónica</h1>
      <h1>Trabajo de Integración Curricular</h1>
      <h2>Estudio, diseño e implementación de una solución para la
recolección de datos provenientes del sistema de riego usando servicios
en la nube y presentación de los datos mediante una página web</h2>
      <h3>Estudiante: Jonathan Alvarez</h3>
      <h3>Director: Ramiro Morejón</h3>
      <div style="text-align: center; margin-top: 20px;">
        <a href="/"><button style="font-size: 25px; padding: 10px 20px;
background-color: #1f6fb1; color: white; border: 2px solid rgb(1, 7, 17);
font-weight: bold;border-radius: 15px;">Ingresar</button></a>
      </div>
    </div>
    <div style="width: 45%; text-align: center;">
      <!-- Your image goes here -->
      <img  src="/static/img/epn.png"  alt="Logo  Escuela  Politecnica
Nacional" width="50%" height="50%">
    </div>
  </div>
{% endblock %}
```

ANEXO V. Código fuente base.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta    name="viewport"    content="width=device-width,    initial-
scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link       rel="stylesheet"       href="{{       url_for('static',
filename='css/main.css') }}">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

    {% block head %}{% endblock %}
</head>

<body>
    {% block body %}{% endblock %}
</body>
</html>
```

ANEXO VI. Código fuente main.css

```
@import
url('https://fonts.googleapis.com/css2?family=Noto+Sans+JP:wght@100;400
;900&display=swap');
/* Tarjetas con datos recientes */
*{
      margin: 0;
      padding: 0;
      box-sizing: border-box;
      font-family: 'Noto Sans JP', sans-serif;
}
body{
      background: #eeeff4;
}
.wrapper .header{
      display: flex;
    width: 100%;
    height: 50px;
    background: #4B9CD3;
    color: #fff;
    font-size: 20px;
    text-transform: uppercase;
    letter-spacing: 5px;
    font-weight: 900;
    align-items: center;
}
.header h2{
    text-align: center;
}
.button-container button {
    background-color: #25445a;
    color: white;
    padding: 8px 16px;
    border-radius: 3px;
    border: 1px solid #7aa7c7;
    cursor: pointer;
    margin-right: 5px;
      font-size: 15px;
      text-align: center;
```

```
}
.wrapper .last_update{
    width: 100%;
      height: 40px;
    color: #4C516D;
    display: flex;
    font-size: 15px;
}
.cards_wrap{
      padding: 10px;
      width: 100%;
      display: flex;
      justify-content: space-between;
      flex-wrap: wrap;
}
.cards_wrap .card_item{
      padding: 15px 15px;
      width: 20%;
}
.cards_wrap .card_inner{
      background: #fff;
      border-radius: 5px;
      padding: 35px 20px;
      min-width: 150px;
      min-height: 200px;
      max-height: 250px;
      width: 100%;
}
.cards_wrap .card_item img{
      width: 80px;
      height: 80px;
      margin-bottom: 5px;
}
.cards_wrap .card_item .variable{
      color: #e36686;
      font-weight: 900;
      letter-spacing: 2px;
      text-transform: uppercase;
      font-size: 20px;
      white-space: nowrap;
```

```
    overflow: hidden;
    text-overflow: ellipsis;
}

.cards_wrap .card_item .nombre_variable{
      color: #04040e;
      font-size: 12px;
      font-weight: 100;
      margin: 5px 0 10px;
}

.cards_wrap .card_item .descripcion{
      font-size: 14px;
      line-height: 24px;
      color: #7b8ca0;
}
/* Donde aparece el grafico */
.chart-container {
    position: relative;
    margin: auto;
    height: 20h;
    width: 60vw;
  }


@media screen and (max-width: 1024px){
      .cards_wrap .card_item{
            width: 33%;
      }
}

@media screen and (max-width: 768px){
      .cards_wrap .card_item{
            width: 50%;
      }
      .wrapper .header{
            font-size: 16px;
            height: 60px;
      }
}
```

```
@media screen and (max-width: 568px){
      .cards_wrap .card_item{
            width: 100%;
      }
      .wrapper .header{
            font-size: 14px;
      }
}

/* CSS para la tabla en datos historicos */
table {
    border: 4px solid black;
    background-color: #f2f2f2;
    font-size: 16px;
    width: 100%;
}
td {
    padding: 8px;
    text-align: center;
    color: #333333;
}
th {
    padding: 8px;
    text-align: center;
    color: #333333;
    font-weight: bold;
}
tr:hover {
    background-color: #f5f5f5;
}
/* Genera los cuadros de las celdas */
td, th {
    padding: 8px;
    text-align: center;
    color: #333333;
    border-bottom: 1px solid #ddd;
    border-left: 1px solid #ddd;
    border-right: 1px solid #ddd;
}
```

ANEXO VII. Código fuente Arduino Yun

```
// Se incluye las librerias necesarias
#include <Process.h> //Libreria de procesos para usar comandos Linux en
Yun Shield
#include <Console.h> //Libreria usada para mostrar informacion en el IDE

// Datos generales
const unsigned long postingInterval = 60000; //Periodo entre envio de la
informacion
unsigned long lastRequest = 0; //Tiempo desde el ultimo request
String dataString = ""; //Donde se guardaran los datos para enviar
String url = "https://invernadero-epn.herokuapp.com/"; //URL del
invernadero
void setup() {
  // Inicio de la consola
  Bridge.begin();
  Console.begin();
  while (!Console); //Espero a la red serial que inicie
  Console.println("Bienvenido al Sistema de Invernadero");
  // Se actualiza los datos llamando a las funciones
  updateData(); //Funcion encargada de actualizar los datos
  sendData(); //Envia los datos hacia la nube
  lastRequest = millis();
}
void loop() {
  // Tomamos una estampa en el tiempo para determinar el tiempo entre
envio
  long now = millis();
  // Se determina si el tiempo que ha pasado es mayor al periodo entre
envio
  if (now - lastRequest >= postingInterval) {
    updateData(); //Funcion encargada de actualizar los datos
    sendData(); //Envia los datos hacia la nube
    lastRequest = now; //Se reinicia el valor para determinar el tiempo
transcurrido
  }
}
```

```
// Esta función genera los valores teniendo en cuenta los rangos investigados
// Luego les da la forma 'variable=valor' seguido de & para completar el cuerpo de la solicitud
// Uso de variable global para almacenarlos datos
void updateData() {
  dataString = "content=EnviadoArduino&";
  dataString += "temperatura=";
  dataString += String(random(14) + 28);
  dataString += "&";
  dataString += "humedad=";
  dataString += String(random(36) + 54);
  dataString += "&";
  dataString += "humsuelo=";
  dataString += String(random(21) + 40);
  dataString += "&";
  dataString += "nivelPH=";
  dataString += String(random(3) + 6);
  dataString += "&";
  dataString += "luminosidad=";
  dataString += String(random(100) + 1);
  Console.print(dataString);
}

// Este metodo hace un HTTP Post request hacia la pagina web para guardar los datos
void sendData() {
  // Se crea un proceso en linux que usara curl para enviar los datos
  Process pr; //Se crea un proceso en linux
  Console.print("\n\nEnviando datos... ");
  pr.begin("curl"); //Se llama al comando curl
  // Se procede a agregar los parametros
  pr.addParameter("-k");
  pr.addParameter("-X");
  pr.addParameter("POST");
  pr.addParameter("-H");
  pr.addParameter("Content-Type: application/x-www-form-urlencoded");
  // Se añade los datos
  pr.addParameter("-d");
  pr.addParameter(dataString);
```

```
  pr.addParameter(url);
  pr.run();
  Console.println("\nDatos del Sistema de Invernadero enviados
exitosamente!");
  // Si se envia datos desde la pagina web se mostraran en consola
  while (pr.available() > 0) {
    char c = p.read();
    Console.write(c);
  }
}
```