Diseño de arquitectura de un software para un prototipo de gestión de inventario con tecnologías en la nube

DANIEL ANDRES VILLA AHMAD

FABIAN STIVEN MORENO ALFONSO

Tutor: WILSON SOTO

Institución Universitaria Politécnico Grancolombiano

Ingeniería de sistemas

Bogotá D.C., Colombia

2023

1. INTRODUCCIÓN

El contexto del problema es el manejo de inventarios en pequeñas y medianas empresas. Actualmente, muchas de estas empresas aún utilizan métodos tradicionales para el manejo de sus inventarios, como el uso de hojas de cálculo o registros en papel. Esto puede ser un proceso lento y propenso a errores, lo que puede resultar en la falta de existencias o la acumulación de productos. Se realizó encuesta a 10 PYMES ubicadas en un sector comercial del municipio de Soacha obteniendo como resultado que todas estas empresas en algún momento de su historia han perdido información importante por tenerla consignada en hojas de papel o cuadernos, representando el 100% de los encuestados se concluye que, si es necesario desarrollar una solución para esta problemática.

La idea de desarrollar este proyecto surgió de la problemática que se evidenció como resultado de las encuestas realizadas a pequeñas y medianas empresas sobre el manejo tradicional de sus inventarios, en donde se mostró que utilizaban hojas de cálculo y registros al papel. Algunos manifestaron que los registros que almacenaban en estas herramientas no garantizaban integridad y/o seguridad de la información, debido a que en cualquier momento ésta se podría perder, además las pequeñas y medianas empresas encuestadas no cuentan con un panorama real de que productos se tienen, en que cantidades, existencia, lo cual hace que se puedan presentar perdidas tanto de información, económicas y de credibilidad con los clientes.

Uno de los casos evidenciados con lo anterior, es el negocio de víveres conocido como "Don Pancho", ubicado en el barrio Ducales de Soacha. Este negocio llevaba sus cuentas en un cuaderno en donde registraba información valiosa acerca de sus productos, ventas y deudores. Pero un día desafortunado, este cuaderno desapareció y con él, toda la

información importante para el negocio. "Don Pancho" sufrió grandes pérdidas económicas y de control, ya que muchas personas que debían dinero no pagaron y generaron pérdidas económicas para sus propietarios.

Con el tiempo, el dueño del negocio solucionó esta problemática haciendo uso de hojas de cálculo de Excel para registrar la información, sin embargo, pronto surgió un nuevo problema: el dueño, ya de edad avanzada, necesitaba contratar a alguien que lo ayudara a administrar y atender el negocio, desafortunadamente, este contrato no resultó como se esperaba.

El dueño comenzó a notar descuadres y pérdidas de productos, y se dio cuenta que el registro de inventario estaba en el mismo libro de Excel al que tenía acceso cualquier persona que usara el computador del negocio. El dueño de "Don Pancho" no se dio cuenta de esto hasta que gran parte de sus productos ya se habían perdido. La persona encargada de administrar el negocio había estado vendiendo productos sin registrarlos en las ventas y quedándose con el dinero.

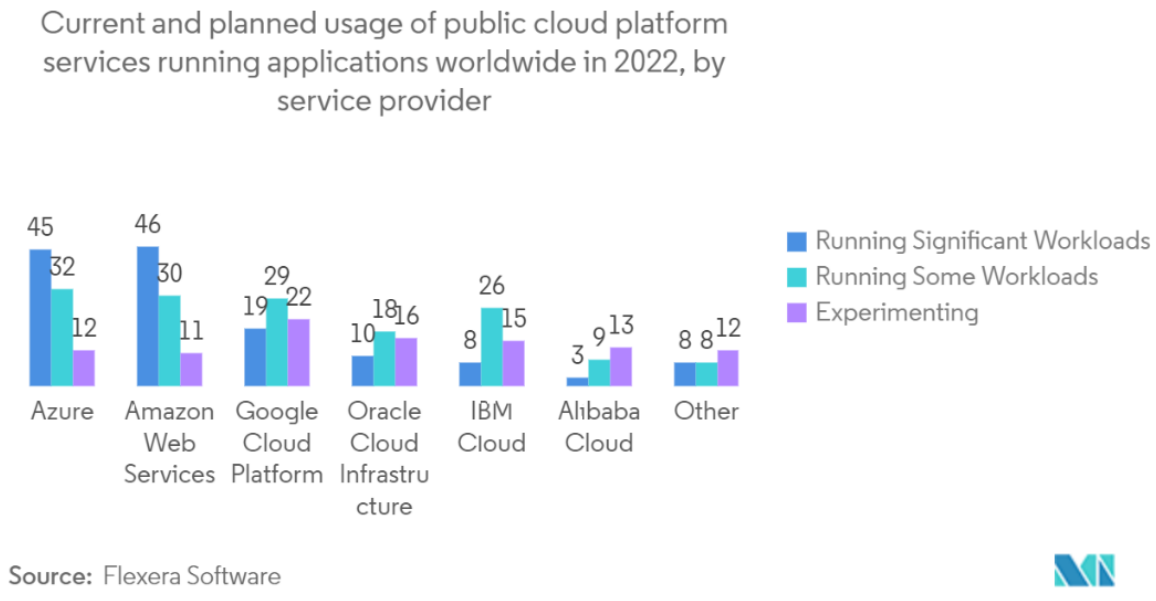
La tensión en "Don Pancho" era cada vez mayor, el dueño se sentía traicionado y vulnerable ante las pérdidas sufridas. Ahora, el negocio estaba en peligro y el futuro de "Don Pancho" parecía incierto. ¿Podrá el dueño recuperar lo perdido? ¿Logrará sacar adelante su negocio después de tantas pérdidas y engaños?

Un sistema de gestión de inventario basado en la nube puede proporcionar a las pequeñas y medianas empresas una solución tecnológica para mejorar el manejo de sus inventarios. Al estar basado en la nube, el sistema permite a los usuarios acceder a la información en tiempo real desde cualquier dispositivo conectado a internet. Además, estos

sistemas suelen ofrecer una amplia gama de funcionalidades, como el seguimiento y control del stock de productos, lo que permite una mejor gestión y control de los inventarios.

Adicionalmente para introducir al proyecto, se ha realizado un estudio de mercado en el cual se obtuvo la siguiente información:

- “Se espera que el tamaño del mercado minorista de la nube crezca de USD 38,46 mil millones en 2023 a USD 104,54 mil millones para 2028, a una CAGR del 22,14 % durante el período de pronóstico (2023-2028)”. (Mordor Intelligence, n/a)
- Según la empresa Flexera Software en una encuesta realizada el año pasado, por lo menos un 46% de los encuestados ya contaban con significativas cargas de trabajo en el proveedor de nube AWS (Amazon Web Services) y el 45% contaba con cargas de trabajo críticas en el proveedor de nube Azure. Por lo cual, con el incremento del uso general de los servicios que ofrecen estos proveedores de nube, el mercado minorista de la nube será testigo de una tasa de crecimiento exponencial durante periodos posteriores.



También en este análisis de mercado se han encontrado múltiples sistemas de gestión de inventarios basado en la nube que pueden llegar a ser competencia directa del prototipo del sistema desarrollado, son las siguientes:

Software inventario	Ideal para...	Ventaja principal	Planes y tarifas*
Dolibarr	Pymes, asociaciones, autónomos.	Fácil y simple utilización.	Gratis
Holded	Pymes.	ERP conectado con otras herramientas.	A partir de 29€
Monstock	Todo tipo de empresas.	Aplicación móvil.	Planes de pago a partir de 14,90€/mes
Oddo Inventory	Todo tipo de empresas.	Alojamiento y soporte gratis.	Gratis
Square	Todo tipo de negocios.	Integración con TPV y otras herramientas.	Gratis
Zoho Inventory	Empresas en crecimiento.	Integración con Zoho CRM y Zoho Books.	Planes de pago a partir de 49€/mes

Fuente: *https://www.appvizer.es/revista/organizacion-planificacion/gestion-de-stock/software-inventario*

2. PROPUESTA

2.1.JUSTIFICACIÓN

La justificación de este proyecto de investigación radica en la necesidad de las pequeñas y medianas empresas (PYMES) de contar con herramientas tecnológicas para mejorar su gestión de inventario. Las PYMES representan un sector importante en la economía, sin embargo, a menudo enfrentan dificultades para competir con grandes empresas que cuentan con sistemas informáticos más avanzados. Un sistema de gestión de inventario basado en la nube puede ayudar a las PYMES a controlar mejor sus niveles de inventario, reducir costos.

Además, el sistema ofrecerá funcionalidades básicas de gestión de inventarios, lo que permitirá a las empresas mantener un mejor control de sus existencias y necesidades.

2.2.ALCANCE

Debido a la limitación de tiempo, el alcance de este proyecto se enfocará en desarrollar un prototipo que ofrecerá funcionalidades básicas de gestión de inventario, incluyendo el seguimiento y control del stock de productos. La interfaz de usuario será intuitiva y fácil de usar para permitir que ellos realicen tareas de gestión sin problemas, además se realizará una implementación de este prototipo funcional sobre alguna de las plataformas o proveedores de nube, para que en un futuro próximo se pueda realizar un software escalable y de alta disponibilidad.

El público objetivo son los propietarios de las pequeñas y medianas empresas que no cuenten con un sistema informático para el manejo de su inventario, cabe señalar que el alcance del proyecto no incluirá la integración del sistema con otros sistemas existentes en la tienda, ni la capacitación del personal que haga parte de estas pequeñas y medianas empresas.

2.3.OBJETIVO PRINCIPAL

Diseñar e implementar una arquitectura para un prototipo de un sistema de gestión de inventario en pequeñas y medianas empresas (PYMES) basado en computación en la nube.

2.4.OBJETIVOS SECUNDARIOS

- Identificar los requerimientos para los procesos de gestión de inventario actuales de las pequeñas y medianas empresas, incluyendo los sistemas y herramientas que se utilizan.
- Diseñar e implementar un sistema prototipo de gestión de inventario que permita el seguimiento y control en tiempo real del stock de productos.
- Desplegar la aplicación en un proveedor de servicios en la nube.
- Comparar el proveedor de servicio en la nube AWS y el proveedor Render.

2.5.PREGUNTA DE INVESTIGACIÓN

¿Se puede construir un sistema de gestión de inventarios para una PYME y se puede desplegar en un proveedor de servicios en la nube?

3. METODOLOGÍA

3.1.DESCRIPCIÓN DE ACTIVIDADES

Para el desarrollo se escogerá scrum como metodología ágil, antes de comentar como se utilizará esta metodología se explicará que es scrum y de que se conforma.

Scrum es un marco de trabajo ágil utilizado en la gestión y desarrollo de proyectos de software. Fue creado para ayudar a los equipos de desarrollo a trabajar de manera más eficiente y efectiva, y se centra en la colaboración, la flexibilidad y la entrega continua de valor al cliente.

En scrum cuando se habla de “Sprint” lo que se quiere decir es que son los ciclos cortos y regulares en los que el equipo trabaja y realiza ciertas entregas en las cuales pueden ser planificación, diseño, desarrollo, prueba y entrega de un conjunto de funcionalidades o características del producto, hay otro aspecto también importante que es “Daily Scrums” este término se define como reuniones diarias en las cuales se reúne el equipo a discutir el proceso y progreso del desarrollo del producto.

El marco de trabajo de Scrum incluye roles definidos, como los siguientes:

- Product Owner
- Scrum Master
- Equipo de desarrollo

Antes de pasar al porque elegimos está, en vez de otras y realizar un cuadro comparativo entre las diferentes metodologías en las que se pensó trabajar, explicaremos como se incluirá esta metodología en el proyecto. Se hizo uso de una herramienta llamada

Jira para realizar el tablero de tareas que se dará a conocer tanto en este segmento como en el cronograma de actividades.

La metodología que se eligió fue Scrum ya que esta tiene ciertos rasgos o características que el proyecto requiere para que esté bien enfocado, estas características son: Flexibilidad, Trabajo en equipo, Mayor control, Mejora la comunicación, Mejora la calidad del producto.

Estas características otorgan ciertos beneficios al proyecto y hacen que todo esté más ordenado, ya que en el rasgo de flexibilidad tenemos lo llamado sprints esto permite que el desarrollo de software sea más ordenado y permite ver tanto avances como obstáculos en el proceso, también se podrían en dado caso que se necesiten ajustes y cambios en el plan durante el proceso.

Otra característica para resaltar es el trabajo en equipo, ya que esto hace que las personas involucradas en el proceso estén continuamente colaborando entre los mismos miembros, esto aumenta la eficiencia y la efectividad del trabajo. Si unimos esta característica con la de mayor control nos da un mayor orden en el equipo, también en el caso en que pueda pasar algo se puede estar atentos y tener una respuesta rápida, ya que esta característica lo que nos permite es realizar las “Daily Scrums” o reuniones diarias para ver progresos del equipo y si hay algún obstáculo y cómo podemos superarlo.

Otra ventaja del trabajo en equipo es que se complementa con la característica de “Mejora la comunicación” ya que Scrum fomenta la comunicación continua entre los miembros del equipo, lo que puede ayudar a identificar y resolver problemas con mayor rapidez.

Y por último tenemos “Mejora la calidad del producto” esto hace referencia a que como se realizan ciertas entregas tempranas y frecuentes del producto en muchos casos permite la mejora en la satisfacción del cliente y en el valor del producto final, también ya que se hacen ciertas entregas permite a los equipos recibir comentarios tempranos y frecuentes sobre el trabajo realizado, lo que les permite ajustar y mejorar continuamente su trabajo.

Otras metodologías de gestión de proyectos que se tuvieron en mente fueron: Waterfall y Kanban además de la anterior nombrada Scrum, estas se tuvieron en mente ya que además de que son conocidas y efectivas también tenían algunos aspectos que nos parecían importantes, a continuación, se dará una breve explicación de estas dos metodologías.

Primero waterfall o modelo en cascada en esta metodología los proyectos se dividen en fases que se deben completar de forma secuencial, estas fases normalmente son: análisis de requisitos, el diseño, la implementación, la verificación y la implementación. Como se dijo es de manera secuencial es decir que una fase no empieza hasta que la anterior no haya se haya dado como terminada.

Ahora Kanban que tiene como objetivo optimizar el flujo de trabajo, identificar cuellos de botella y eliminar el trabajo innecesario para maximizar la eficiencia del equipo, esta metodología normalmente se trabaja mediante un tablero en el cual hay diferentes tareas que pasan por diferentes etapas que van desde “Por Hacer” el cual es un estado inicial de la mayoría de tareas hasta una etapa “Hecho” o “Realizado” en el cual se llega cuando la tarea ya está culminada y se puede pasar a la siguiente tarea, cabe resaltar que estas tareas por lo general van siendo asignadas individualmente.

A continuación, se presentará un cuadro comparativo entre Waterfall, Kanban y Scrum para así demostrar sus diferentes características en diferentes campos que aplican a las 3 y así poder concluir por qué se eligió la metodología Scrum y demostrar su progreso con la aplicación para la gestión “Jira”:

Característica	**Waterfall**	**Kanban**	**Scrum**
Enfoque	Secuencial	Evolutivo	Iterativo e incremental
Planificación	Planificación detallada al principio del proyecto	Planificación continua según las necesidades del proyecto	Planificación al inicio de cada iteración
Entrega de valor	Al final del proyecto	Continua y en pequeñas entregas	Al final de cada iteración
Gestión de cambios	Difícil de implementar cambios	Flexibilidad para hacer cambios en cualquier momento	Flexibilidad para hacer cambios en cada iteración
Roles	Equipo de proyecto, liderado por un gerente de proyecto	No hay roles fijos, el equipo se auto organiza	Equipo de desarrollo, propietario de producto y facilitador
Tiempo de ciclo	Largo	Variable	Corto
Comunicación	Poca colaboración y comunicación durante el proceso	Comunicación continua y colaboración entre los miembros del equipo	Comunicación continua y colaboración entre los miembros del equipo

Enfoque en la calidad	La calidad se verifica al final del proyecto	La calidad se verifica durante todo el proceso	La calidad se verifica al final de cada iteración

Tabla 1. *Diferencias entre Waterfall, Kanban, Scrum*

Fuente: *Diferentes recursos*

Al estudiar y comparar las diferentes metodologías ágiles, concluimos que para este proyecto se elige la metodología scrum, teniendo en cuenta los beneficios, ventajas y características que esta nos ofrece.

Los Sprint con sus MVP serán:

- **Sprint 1**: En este vamos a estudiar las diferentes posibilidades que tenemos en el mercado, esto para escoger una problemática a solucionar y un nicho de mercado como público objetivo. **MVP**: Pregunta problema planteada y público objetivo identificado.
- **Sprint 2**: Al seleccionar una problemática y tener claro cuál es el público objetivo, se van a levantar los requerimientos funcionales y no funcionales en forma de Historias de usuario. **MVP**: Historias de usuario documentadas para su desarrollo.
- **Sprint 3:** Se estudiarán las diferentes tecnologías que se adaptan a las necesidades específicas del proyecto y en este sprint se empezará con el desarrollo funcional del proyecto. **MVP**: Funcionalidad de inicio de sesión y registro para los usuarios.
- **Sprint 4**: Se definió este sprint donde se hará el desarrollo completo de las funcionalidades del sistema dando cumplimiento a las historias de usuario planteadas en el sprint 2. **MVP**: Proyecto completo en funcionalidades y funcionando en un entorno local.

- **Sprint 5**: Se estudiarán los proveedores de nube (Amazon Web Service y Render.com), haciendo un estudio de sus capas gratuitas y sus limitaciones. **MVP**: Listados de limitaciones y restricciones de la capa gratuita de los proveedores estudiados.
- **Sprint 6**: Luego de estudiar y validar las ventajas y desventajas de cada uno de los proveedores, los cuales nos debe permitir la capa gratuita realizar el despliegue del proyecto haciendo uso de sus servicios. **MVP**: Proyecto desplegado en 1 de los proveedores de nube.
- **Sprint 7**: Se desplegará el proyecto en el segundo proveedor de nube y tabla de diferencias entre los dos proveedores. MVP: Proyecto desplegado en los 2 proveedores de nube y tabla de diferencias entre ambos.

3.2.CRONOGRAMA DE ACTIVIDADES

A continuación, se mostrarán las diferentes actividades tanto realizadas como por realizar en la aplicación comentada anteriormente llamada Jira:

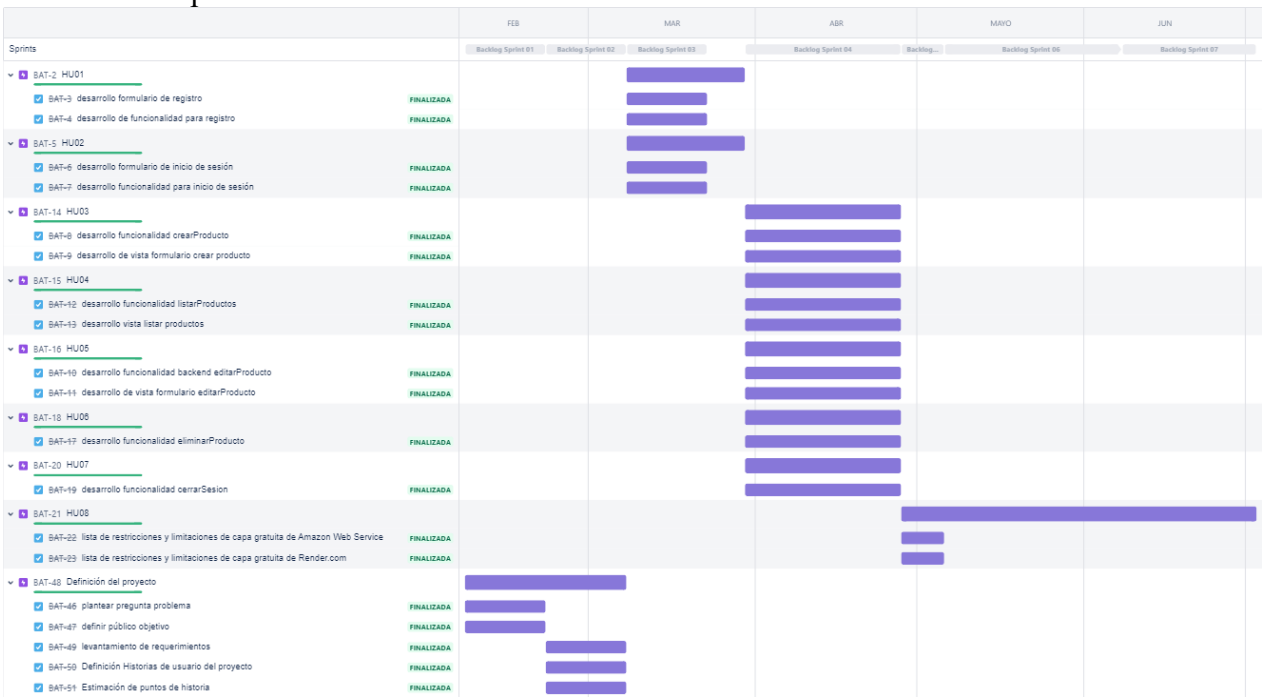


Imagen 1. *Cronograma Scrum sprint 1 – sprint 5*

Fuente: *Jira - Elaboración propia*

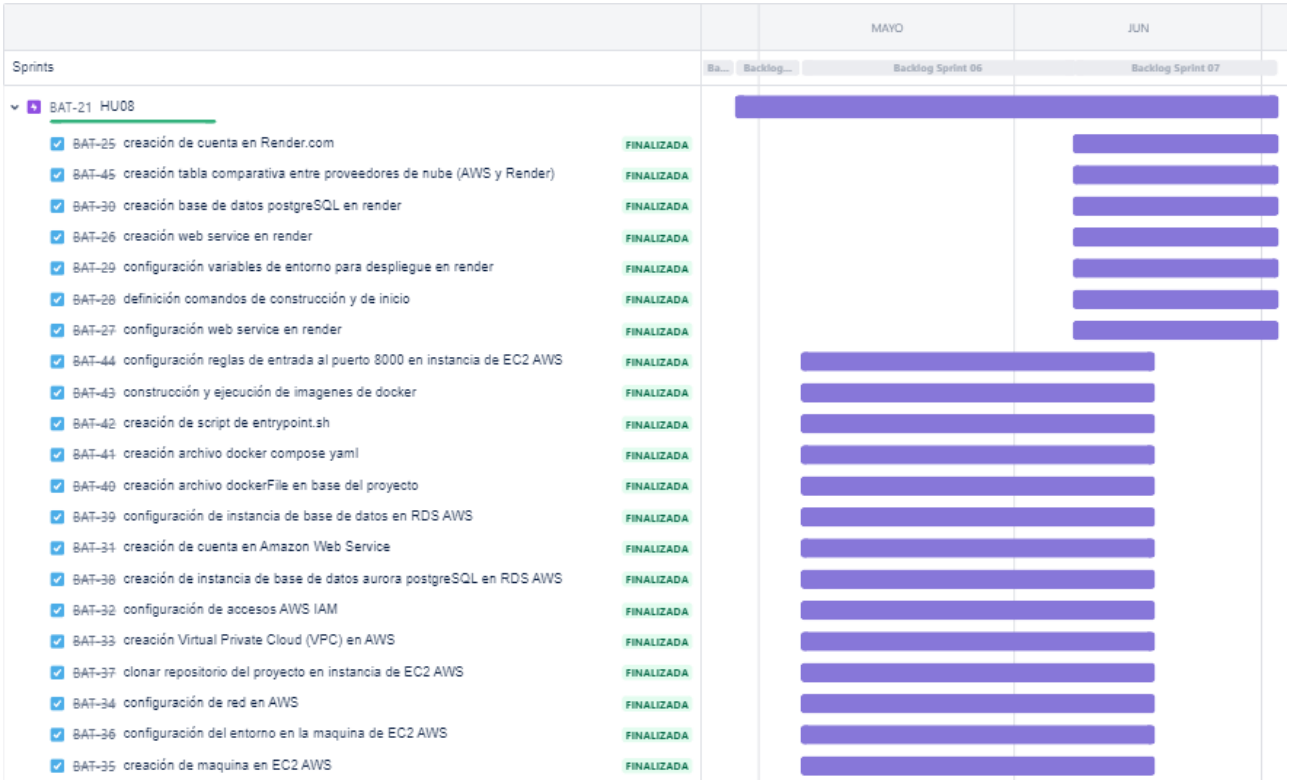
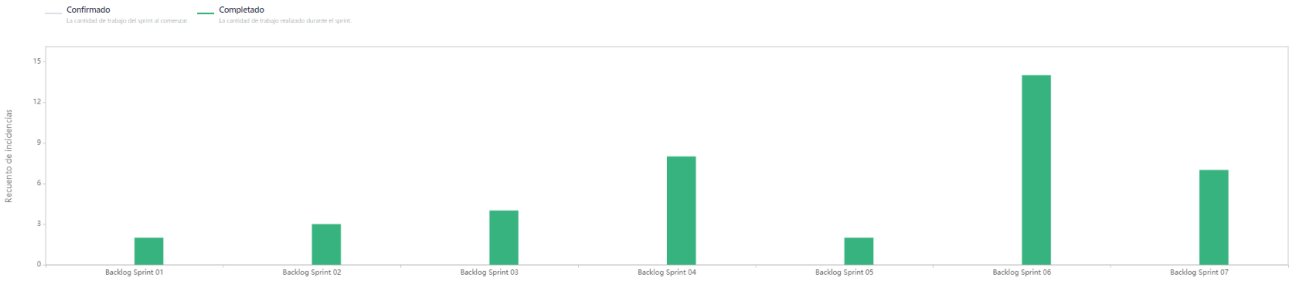


Imagen 2. *Cronograma Scrum sprint 6 – sprint 7*

Fuente: *Jira - Elaboración propia*

En estas imágenes se puede apreciar cada una de las tareas correspondientes a cada uno de los Sprints planteados y con su respectiva HU relacionada, cabe recalcar que al utilizar una metodología ágil los cambios que se requieran se manejarán como controles de cambio que serán ejecutados durante el sprint en el que se hayan solicitado.

3.3.INFORME DE VELOCIDAD



Sprint	Confirmado	Completado
Backlog Sprint 01	0	2
Backlog Sprint 02	0	3
Backlog Sprint 03	0	4
Backlog Sprint 04	0	8
Backlog Sprint 05	0	2
Backlog Sprint 06	0	14
Backlog Sprint 07	0	7

Imagen 3. *Informe velocidad - Tareas/Sprint*

Fuente: *Jira*

4. MARCO TEORICO

Como insumo para la ejecución del proyecto se tuvo en cuenta las siguientes consideraciones teóricas, las cuales tendrán una secuencia lógica enfocada al ciclo de vida de un proyecto.

4.1. GENERALES

4.1.1. GESTIÓN DE INVENTARIOS

"La gestión de inventarios es un sistema utilizado principalmente en la industria minorista para seguir el movimiento de las materias primas y los productos acabados."(SafetyCulture, 2023)

Esta gestión de inventarios generalmente se divide en dos tipos los cuales son el sistema de inventario periódico y el perpetuo, la única diferencia que tienen estos dos es que un inventario se actualiza periódicamente y el otro tipo se actualiza en tiempo real. Cuando una empresa va a elegir uno de los dos tipos tiene que ver qué tipo de empresa, el tamaño y el tipo de inventario.

"El control de inventarios es un elemento clave de un sistema de gestión de inventarios. Los jefes de almacén y los planificadores de la producción deben seguir las siguientes actividades y procedimientos para controlar su inventario:" (SafetyCulture, 2023)

1. Recepción, almacenamiento y traslado de mercancías
2. Colocación de artículos en lugares estratégicos
3. Seguimiento de los artículos del inventario y su ubicación en el almacén
4. Documentar los detalles e historiales de los productos
5. Controlar el estado de los artículos en stock
6. Cumplir con los pedidos de compra con el stock disponible

7. Integración de escáneres de códigos de barras
8. Formación de informes de reordenación

Hay ciertas técnicas o métodos que ayudan a este control de inventarios algunas de ellas son:

Análisis ABC:

1. Clase A: artículos caros con controles estrictos
2. Clase B: artículos de precio medio con un volumen de ventas y unas existencias medias
3. Clase C: artículos de bajo valor con grandes ventas y enormes inventarios

LIFO y FIFO

"Con el **método LIFO**, el almacén distribuye primero el lote más reciente de artículos a los clientes. Pero con la **técnica FIFO**, el almacén da prioridad a las existencias más antiguas para su procesamiento y envío." (SafetyCulture, 2023)

4.1.2. DJANGO

Django es un framework que se utiliza para el desarrollo de aplicación web usando Python como lenguaje de programación. "Django considera algunas funcionalidades listas para usar para facilitar el desarrollo de aplicaciones web. Como resultado, no es necesario escribir todo el código ni usar tiempo para buscar errores de código en el framework" (Vidal-Silva et al., 2021). Es decir, mediante Django, el desarrollo de sistemas de información web puede ser rápido, seguro, escalable y también fáciles de mantener (Salas-Zárate et al., 2015).

Por parte del apartado de Django nos indica unas ventajas las cuales son:(Django, s/f)

1. Rápido: Django fue diseñado para ayudar a los desarrolladores a plasmar sus ideas con mayor facilidad
2. Escalable: Algunos de los sitios más visitados utilizan Django ya que escala de forma rápida y flexible.
3. Versátil: Se pueden crear desde sistemas de gestión, redes sociales hasta plataformas informáticas científicas.

4.1.3. REST API

En el contexto de un sistema de inventarios basado en la nube para pequeñas y medianas empresas (PyMEs), REST API (Application Programming Interface) se refiere a una interfaz de programación de aplicaciones basada en el protocolo HTTP que permite la comunicación entre diferentes sistemas informáticos a través de internet (Fielding, 2000). En este sentido, REST API se convierte en una herramienta clave para la integración de diferentes componentes de software en un sistema de inventarios basado en la nube.

REST API permite a los desarrolladores de software acceder a los datos y funcionalidades de un sistema de inventarios basado en la nube mediante solicitudes HTTP, lo que reduce significativamente la complejidad del desarrollo y la implementación de software. Además, REST API es un estándar abierto y ampliamente utilizado en la industria del software, lo que aumenta la interoperabilidad y la capacidad de integración de diferentes sistemas. (Khattak, Ali & Khan, 2016)

Por otra parte, la implementación de una REST API en un sistema de inventarios basado en la nube para PyMEs puede mejorar significativamente la eficiencia y la escalabilidad del sistema. REST API permite a los usuarios acceder a los datos y funcionalidades del sistema

desde cualquier dispositivo con conexión a internet, lo que aumenta la movilidad y la flexibilidad de la gestión de inventarios. (Lee & Kim,2019) Además, REST API permite a los sistemas informáticos procesar solicitudes de manera más eficiente y escalable que los métodos tradicionales basados en la integración de sistemas mediante middleware.

En conclusión, la implementación de una REST API en un sistema de inventarios basado en la nube para PyMEs puede mejorar significativamente la eficiencia y la escalabilidad del sistema, además de aumentar la interoperabilidad y la capacidad de integración de diferentes sistemas informáticos.

Adicionalmente, La computación en la nube es un modelo de distribución de recursos de cómputo que permite el acceso bajo demanda a través de una red de comunicaciones, típicamente internet. Este modelo ha tenido un gran impacto en la industria, ya que ha permitido a las empresas reducir los costos de infraestructura y aumentar la eficiencia operativa. La computación en la nube se basa en tres características esenciales: elasticidad, servicio bajo demanda y acceso por red. (Chou et al., 2012)

4.1.4. ELASTICIDAD

La elasticidad se refiere a la capacidad de la infraestructura de cómputo para escalar hacia arriba o hacia abajo según las necesidades de la empresa. Esto permite a las empresas ahorrar costos y evitar la sobrecarga de recursos. El servicio bajo demanda significa que los recursos de cómputo se pueden adquirir y liberar según sea necesario, lo que permite una mayor flexibilidad en la gestión de los recursos de la empresa. El acceso por red significa que los recursos de cómputo están disponibles a través de una red de comunicaciones, lo que permite a las empresas acceder a ellos desde cualquier lugar con una conexión a internet.

En el contexto de un sistema de inventarios para pequeñas y medianas empresas, la computación en la nube permite a estas empresas acceder a una infraestructura de cómputo de alta calidad sin tener que invertir en la infraestructura de cómputo y en la contratación de personal especializado para gestionarla. Esto significa que las empresas pueden centrarse en su negocio principal sin preocuparse por la gestión de la infraestructura de cómputo. Además, la computación en la nube permite a las empresas escalar su infraestructura de cómputo según sus necesidades, lo que les permite ahorrar costos y evitar la sobrecarga de recursos.

4.1.5. COMPUTACIÓN EN LA NUBE

En la actualidad, el uso de la tecnología en la gestión empresarial es fundamental para mejorar la eficiencia y eficacia de los procesos. En particular, el uso de la computación en la nube se ha convertido en una herramienta valiosa para las empresas, especialmente para las pequeñas y medianas empresas (PyMEs), ya que les permite acceder a recursos tecnológicos sin tener que invertir grandes cantidades de dinero en infraestructura de TI.

El sistema de inventarios es uno de los procesos críticos en la gestión empresarial, ya que permite a las empresas gestionar y controlar el stock de productos y materiales, lo que a su vez influye en la satisfacción del cliente y en la rentabilidad de la empresa. En este sentido, el uso de un sistema de inventarios basado en la nube puede ofrecer múltiples beneficios para las PyMEs, como la reducción de costos, la mejora de la eficiencia en la gestión de inventarios y la facilidad de acceso a la información.

4.2. DIAGRAMAS

4.2.1. DIAGRAMAS UML

UML (Unified Modeling Language) o lenguaje estandarizado de modelado. “Está especialmente desarrollado para ayudar a todos los intervinientes en el desarrollo y modelado de

un sistema o un producto software a describir, diseñar, especificar, visualizar, construir y documentar todos los artefactos que lo componen, sirviéndose de varios tipos de diagramas.”(DiagramasUML, s/f)

Hay diferentes tipos de diagramas UML y cada uno tiene sus propios subtipos, algunos de los más usados: (Omg, 2009)

1. Diagramas estructurales
    - Diagrama de clases: Muestran la estructura del sistema utilizando las clases con sus características, dependencias, etc.
    - Diagrama de despliegue: Muestra la arquitectura del sistema
2. Diagramas de comportamiento
    - Diagrama de caso de uso: Describe las acciones que tienen los diferentes actores que pertenecen al sistema
    - Diagrama de máquina de estados: Se utiliza para modelar el comportamiento de las transiciones de estado finitos

4.2.2. DIAGRAMA DE INFRAESTRUCTURA

Un diagrama de infraestructura en la nube es una representación gráfica de la estructura tecnológica que soporta un sistema basado en la nube. Este diagrama muestra los diferentes componentes que conforman la infraestructura, como servidores, redes, dispositivos de almacenamiento y aplicaciones, así como la interacción entre ellos. Además, un diagrama de infraestructura en la nube también puede mostrar la ubicación geográfica de los componentes y cómo se relacionan con otros servicios en la nube.

En el caso de un sistema de inventarios para PYMES basado en computación en la nube, el diagrama de infraestructura en la nube puede incluir los siguientes componentes:

- Servidores: los servidores son los componentes clave en cualquier sistema basado en la nube. En el caso de un sistema de inventarios para PYMES, los servidores pueden ser utilizados para almacenar y procesar datos relacionados con los inventarios, como los registros de entrada y salida de productos, la cantidad de existencias y la ubicación de estos.
- Redes: las redes son los medios de comunicación que conectan los diferentes componentes de la infraestructura en la nube. En el caso de un sistema de inventarios para PYMES, las redes pueden ser utilizadas para transmitir datos relacionados con los inventarios desde los servidores hasta los dispositivos móviles o de escritorio de los usuarios.
- Dispositivos de almacenamiento: los dispositivos de almacenamiento son utilizados para almacenar los datos relacionados con los inventarios. En un sistema de inventarios para PYMES basado en computación en la nube, los datos pueden ser almacenados en diferentes tipos de dispositivos, como discos duros, unidades flash o en la nube.
- Aplicaciones: las aplicaciones son programas informáticos diseñados para realizar tareas específicas. En el caso de un sistema de inventarios para PYMES, las aplicaciones pueden ser utilizadas para gestionar los inventarios, generar reportes, enviar alertas de bajo inventario, entre otras funciones.

En resumen, un diagrama de infraestructura en la nube es una herramienta útil para entender cómo funciona un sistema de inventarios para PYMES basado en computación en la nube. Este diagrama muestra los diferentes componentes que conforman la infraestructura y

cómo se relacionan entre sí para proporcionar un servicio eficiente y seguro para la gestión de inventarios.

4.2.3. DIAGRAMA DE DESPLIEGUE

Un diagrama de despliegue es una representación visual de la arquitectura de un sistema que muestra cómo los diferentes componentes del sistema se distribuyen en el hardware y el software. En el caso de un sistema de inventarios en la nube para Pymes, el diagrama de despliegue mostraría cómo los diferentes componentes del sistema están distribuidos en la nube, es decir, en los servidores y aplicaciones web que se utilizan para almacenar y gestionar la información del inventario.

En este sentido, el diagrama de despliegue de un sistema de inventarios en la nube para Pymes puede incluir componentes como el servidor de aplicaciones, la base de datos en la nube, los servicios web para la integración con otras aplicaciones y la interfaz de usuario que permite a los usuarios acceder y gestionar la información del inventario. Además, el diagrama de despliegue también debe mostrar cómo se comunican estos componentes entre sí y cómo se accede a ellos desde diferentes dispositivos, como computadoras de escritorio, laptops y dispositivos móviles.

4.3. AMAZON

4.3.1. DOCKER

"Docker es una plataforma de software que le permite crear, probar e implementar aplicaciones rápidamente”(Amazon Web Services, 2023a). Este software lo empaqueta en contenedores donde se incluye todo lo necesario para que el software pueda ser ejecutado, esto incluye bibliotecas, códigos y tiempo de ejecución de este.

"La ejecución de Docker en AWS les ofrece a desarrolladores y administradores una manera muy confiable y económica de crear, enviar y ejecutar aplicaciones distribuidas en cualquier escala."(Amazon Web Services, 2023a)

Docker se utiliza más que todo en estos casos:

- Ejecución de arquitecturas de microservicios distribuidos
- Implementación de código con canalizadores de integración
- Creación de sistemas de procesamiento de datos altamente escalables

4.3.2. DOCKER COMPOSE

Docker compose es una herramienta de plataforma dedicada a la instrumentación local de Dockers, es decir. se utiliza para definir y ejecutar aplicaciones Docker de manera fácil y rápida desde diferentes contenedores.

Para funcionar emplea un archivo tipo YAML el cual permite configurar diferentes servicios que tiene esta aplicación. Algunas de sus características son: (KeepCoding Team, 2023)

- Multiplicidad de entornos en un solo host
- Conservar los datos de volumen
- Recrear contendores que hayan sido modificados
- Variables y movimientos de una composición entre ambientes

4.3.3. LINUX

Linux es un sistema operativo open source que está compuesto por kernel el cual es la interfaz fundamental entre el hardware de una computadora y sus procesos, Linux fue creado por Linus Torvalds en 1991. Este sistema operativo fue lanzado en virtud de la licencia publica general de GNU, es decir, que cualquier persona puede ejecutarlo, estudiarlo y modificarlo.

“Linux sirve como base para casi cualquier tipo de iniciativa de TI, lo cual incluye los contenedores, las aplicaciones desarrolladas en la nube y la seguridad.”(Red Hat, 2018), por ejemplo, en servicios de la nube como Amazon web services o Google cloud plataform ofrecen varias distribuciones de Linux en sus imágenes disponibles al publico

4.3.4. SUBREDES

Una subred es una red dentro de una red. Estas subredes son diseñadas para mejorar la eficiencia de las redes, es decir, el tráfico de la red puede recorrer una distancia más corta sin pasar por routers innecesarios así llegando a su destino.

“Al igual que pasa con el servicio de correos, las redes son más eficientes cuando los mensajes viajan de la forma más directa posible. Cuando una red recibe paquetes de datos de otra red, los clasifica y enruta por subredes para que los paquetes no tomen una ruta ineficiente hacia su destino.”(Cloudflare, 2023)

4.3.5. ZONAS DE DISPONIBILIDAD

Zonas de disponibilidad tiene uno o más centros de datos independientes con alimentación, red y conectividad redundantes en una región de AWS. Estas ofrecen a los usuarios operen bases de datos y aplicación con un nivel de disponibilidad, mayor tolerancia a fallos y escalabilidad que la que proporcionaría un solo centro de datos. (Amazon Web Services, 2023)

Todas las zonas de disponibilidad de una región seleccionada en AWS están interconectadas con redes de gran ancho de banda y baja latencia a través de fibra metropolitana dedicada totalmente redundante para disponibilidad de red de alto rendimiento y baja latencia entre zonas.

4.3.6. CIDR

CIDR, o Classless Inter-Domain Routing, es un método de asignación de direcciones IP que mejora la eficiencia del enrutamiento de datos en Internet. Permite la asignación flexible y eficiente de direcciones IP en una red utilizando máscaras de subred de longitud variable (VLSM). Antes de CIDR, las direcciones IP se asignaban mediante un sistema de direccionamiento basado en clases, que era limitado e ineficiente.

CIDR permite el uso de direcciones IP sin clase, lo que proporciona una mayor flexibilidad y eficiencia en la asignación de direcciones y el enrutamiento de datos.

4.3.7. ACL

Las Listas de Control de Acceso (ACL) son una opción utilizada para gestionar el acceso a buckets y objetos en Amazon Web Services (AWS). Con las ACL, se pueden otorgar permisos básicos de lectura o escritura a otras cuentas de AWS. Sin embargo, existen limitaciones en la administración de permisos con las ACL.

4.3.8. VPC

Amazon Virtual Private Cloud (Amazon VPC) es una red virtual aislada que permite lanzar recursos de AWS en una infraestructura escalable. Funciona de manera similar a una red tradicional en un centro de datos, pero con los beneficios de la nube de AWS. Con Amazon VPC, se pueden configurar subredes, asignar direcciones IP, establecer enrutamiento y utilizar puertas de enlace y puntos de conexión para conectar la VPC a otras redes. También se pueden habilitar características como la replicación de tráfico, los logs de flujo de VPC y las conexiones de VPN. El uso de Amazon VPC no tiene un costo adicional, pero se aplican cargos por componentes adicionales como puertas de enlace NAT y servicios de análisis.

4.3.9. EC2

Amazon EC2, también conocido como Amazon Elastic Compute Cloud, es un servicio de computación en la nube proporcionado por Amazon Web Services (AWS) que ofrece escalabilidad y flexibilidad a los usuarios. Permite desarrollar y desplegar aplicaciones sin necesidad de invertir en hardware inicialmente. Con EC2, es posible lanzar servidores virtuales llamados instancias según sea necesario, configurar la seguridad y las redes, y administrar el almacenamiento. Esta capacidad de escalar hacia arriba o hacia abajo permite adaptarse a cambios en los requisitos o picos de demanda, lo que reduce la necesidad de predecir el tráfico.

4.3.10. TABLAS DE ENRUTAMIENTO

Las tablas de enrutamiento son conjuntos de reglas que determinan la dirección del tráfico de red desde una subred o puerta de enlace. Hay diferentes conceptos asociados a las tablas de enrutamiento, como la tabla de enrutamiento principal, tablas de enrutamiento personalizadas, destino, asociación de tabla de enrutamiento, ruta local, propagación, entre otros.

Cada subred en una VPC debe estar asociada a una tabla de enrutamiento, ya sea de forma explícita o implícita. Las tablas de enrutamiento contienen rutas que especifican el destino y objetivo del tráfico. Por ejemplo, se puede agregar una ruta a una tabla de enrutamiento de una subred para permitir que acceda a Internet a través de una puerta de enlace de Internet.

4.3.11. INTERNET GATEWAY

Internet Gateway es un componente de la Virtual Private Cloud (VPC) de AWS que permite la comunicación entre la VPC y Internet. Funciona como un punto de entrada y salida para el tráfico de red entre los recursos de la VPC y el Internet público.

Una Internet Gateway admite tanto el tráfico IPv4 como el tráfico IPv6. Permite que los recursos en las subredes públicas de la VPC se conecten a Internet si tienen una dirección IP

pública correspondiente. Del mismo modo, los recursos de Internet pueden iniciar una conexión con los recursos de la subred utilizando la dirección IP pública o IPv6.

4.3.12. GRUPOS DE SEGURIDAD

Los grupos de seguridad en AWS funcionan como firewalls que controlan el tráfico que entra y sale de los recursos de una Virtual Private Cloud (VPC). Puedes elegir los puertos y los protocolos para permitir o restringir el tráfico.

4.3.13. BALANCEADOR DE CARGA

Un balanceador de carga orientado hacia Internet tiene un nombre DNS que puede resolverse públicamente, lo que le permite redirigir las solicitudes de los clientes a través de Internet hacia las instancias EC2 que están registradas en el balanceador de carga. Si un balanceador de carga se encuentra en una VPC con ClassicLink habilitado, sus instancias pueden estar vinculadas a instancias EC2-Classic. Por otro lado, si un balanceador de carga está en EC2-Classic, sus instancias deben estar también en EC2-Classic.

4.3.14. AMAZON AURORA SERVERLESS

Amazon Aurora Serverless es una opción de escalado automático bajo demanda para Amazon Aurora, lo cual permite ejecutar bases de datos en la nube sin la necesidad de gestionar instancias de base de datos. Con Aurora Serverless, la capacidad de la base de datos se ajusta automáticamente según las necesidades de la aplicación, lo que resulta en un uso eficiente de los recursos y un ahorro de costos significativo. Aurora Serverless v2 es altamente escalable y puede manejar cargas de trabajo variables, impredecibles y empresariales, ofreciendo características como alta disponibilidad, rentabilidad y facilidad de uso en la administración de la capacidad de la base de datos.

5. RESULTADOS

5.1.ENTREGABLES

Una vez finalizado el proyecto se espera contar con:

- Historias de usuario
- Documentos o diagramas realizados
- Repositorio utilizado en desarrollo
- Backend desplegado en un servicio de la nube
- Pruebas funcionales realizadas

Todo esto también estará subido en el tablero de Scrum que fue la metodología escogida

Se utilizo como plataforma GitHub para subir el repositorio el cual contiene el backend con sus configuraciones para el despliegue: https://github.com/Villah0/TESIS y https://github.com/Villah0/TrabajoGrado/tree/feat/application_aws para AWS

5.2.HISTORIAS DE USUARIO

En estas historias se pueden validar las historias de usuario levantadas a partir de los requerimientos, con la ayuda de Jira se han manejado estas historias de usuario como Épicas, aparte de las historias de usuario también se ha definido una etapa que es la definición del proyecto el cual no es una historia de usuario si no una etapa previa a todas las HU:

HU01

Descripción
Dada: La petición del Usuario **Cuando**: Se quiera registrar

Entonces: Se le redirige

Debe tener:

- Nombre de usuario
- Contraseña

Criterios de aceptación:

- En los campos para el registro del usuario se permitirán caracteres alfanuméricos
- La interfaz de usuario tiene que ser intuitivo
- El botón para registrar usuario debe tener un color azul
- Colores estándar: blanco y negro

HU02

Descripción

Dada: La petición del Usuario
Cuando: Ingrese a su sesión
Entonces: Se le redirige
Debe tener:

- Correo o nombre de usuario
- Contraseña

Criterios de aceptación:

- En los campos para el inicio de sesión del usuario se permitirán caracteres alfanuméricos
- La interfaz de usuario tiene que ser intuitivo
- El botón para iniciar sesión debe tener un color azul
- Colores estándar: blanco y negro

HU03

Descripción

Dada: La petición del Usuario
Cuando: Registrar los productos
Entonces: se le pedirá información
Debe tener:

- Nombre del producto
- Cantidad del producto
- Valor del producto
- Descripción
- Que el sistema le asigne de manera automática un ID a cada producto.

Criterios de aceptación:

- La interfaz de registro de producto tiene que ser intuitivo
- El botón para registro de producto debe tener un color azul
- Colores estándar: blanco y negro
- El campo cantidad y valor solo permitirá números
- El campo descripción será opcional con una cantidad de 255 caracteres

HU04

Descripción

Dada: La petición del usuario

Cuando: Se requiera revisar la información de productos

Entonces: Se listarán todos los productos asociados a ese usuario

Debe tener:

- Nombre del producto
- Cantidad del producto
- Valor del producto
- Descripción
- Que el sistema le asigne de manera automática un ID a cada producto.

Criterios de aceptación:

- La interfaz de listado de productos tiene que ser intuitivo
- El botón para la edición del producto debe tener un color amarillo
- Colores estándar: blanco y negro

- Los productos se mostrarán en una tabla

HU05

Descripción

Dada: La petición del usuario

Cuando: Se requiera editar la información de productos

Entonces: Se precargarán los datos en un formulario para editar la información

Debe tener:

- Nombre del producto
- Cantidad del producto
- Valor del producto
- Descripción

Criterios de aceptación:

- La interfaz de edición de producto tiene que ser intuitivo
- El botón para editar producto debe tener un color amarillo
- El botón para eliminar producto debe tener un color rojo
- Colores estándar: blanco y negro
- El campo cantidad y valor solo permitirá números
- El campo descripción será opcional con una cantidad de 255 caracteres

HU06

Descripción

Dada: La petición del usuario

Cuando: Se requiera eliminar la información de productos

Entonces: Se permitirá al usuario eliminarlo y se eliminará el registro de la base de datos

Debe tener:

Botón de eliminar

Criterios de aceptación:

- La interfaz de eliminar de producto tiene que ser intuitivo
- El botón para editar producto debe tener un color amarillo
- El botón para eliminar producto debe tener un color rojo
- Colores estándar: blanco y negro
- El campo cantidad y valor solo permitirá números
- El campo descripción será opcional con una cantidad de 255 caracteres

HU07

Descripción

Dada: La petición del usuario

Cuando: Cierre sesión

Entonces: Se dirige

Debe tener:

- Botón de cierre de sesión

Criterios de aceptación:

- La interfaz del prototipo tiene que ser intuitivo
- El sistema tendrá una navbar que tendrá los diferentes elementos del sistema o vistas y el cierre de sesión
- Colores estándar: blanco y negro

HU08

Descripción

Como usuario quiero que el sistema esté desplegado con algún proveedor de nube para que esté disponible siempre que se requiera

Dada: La petición del usuario

Cuando: Se quiera ingresar al sistema

Entonces: Este debe estar disponible para ser accedido

Debe tener:

- Despliegue haciendo uso de algún proveedor de nube

Criterios de aceptación:

- El sistema desplegado debe contar con una escalabilidad aceptable
- El proveedor de nube tiene que contar con una capa gratuita para optimizar los recursos
- El sistema desplegado se podrá acceder en cualquier momento y en cualquier dispositivo como: Celular, computador, Tablet, etc
- El proveedor de nube cuente con los servicios necesarios para realizar la implementación del sistema

5.3.DOCUMENTOS

5.3.1 Diagrama clases del sistema

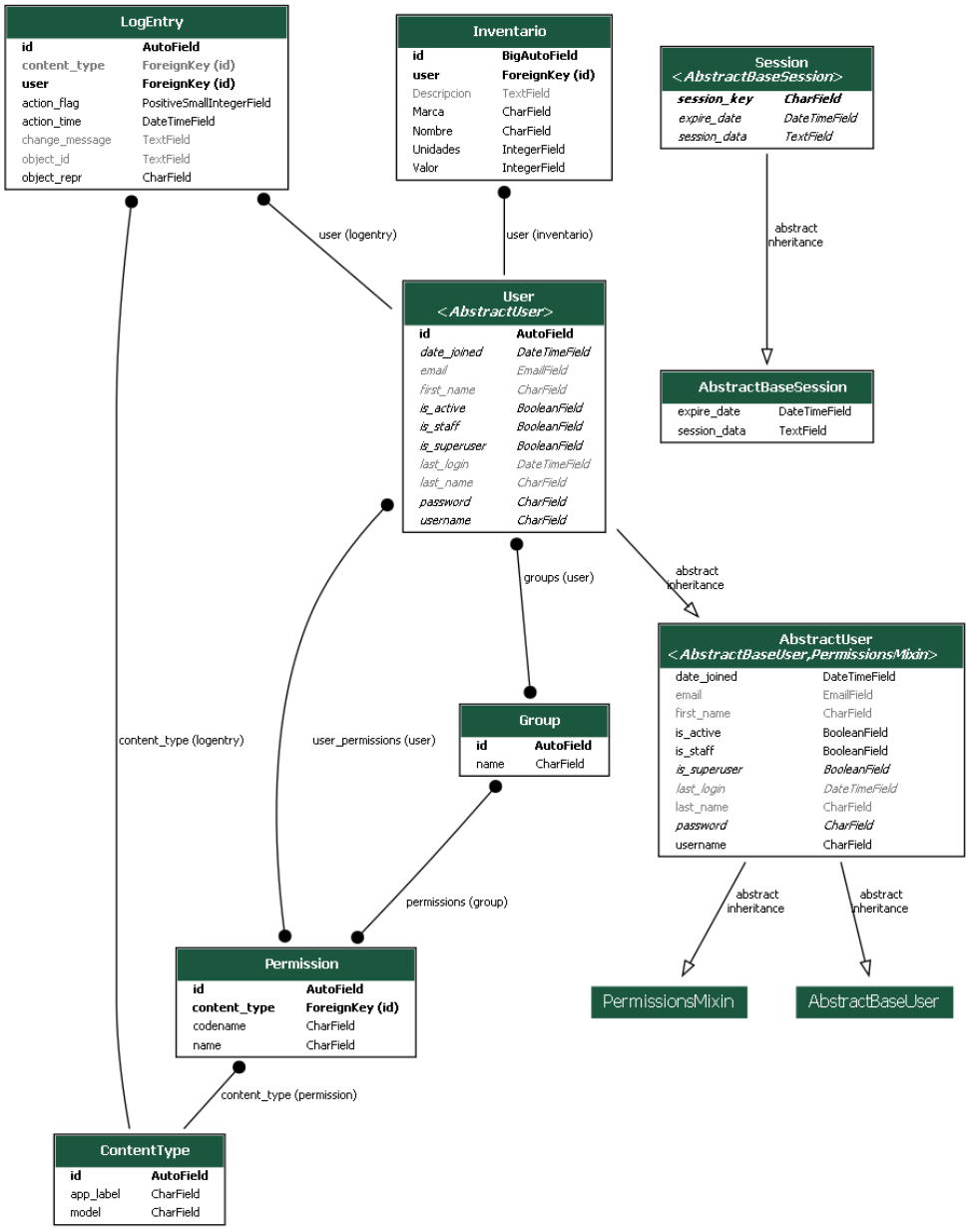


Imagen 4. *Diagrama de clases*

Fuente: *Generación mediante librería pydotplus*

5.3.2 Diagrama funcional del sistema

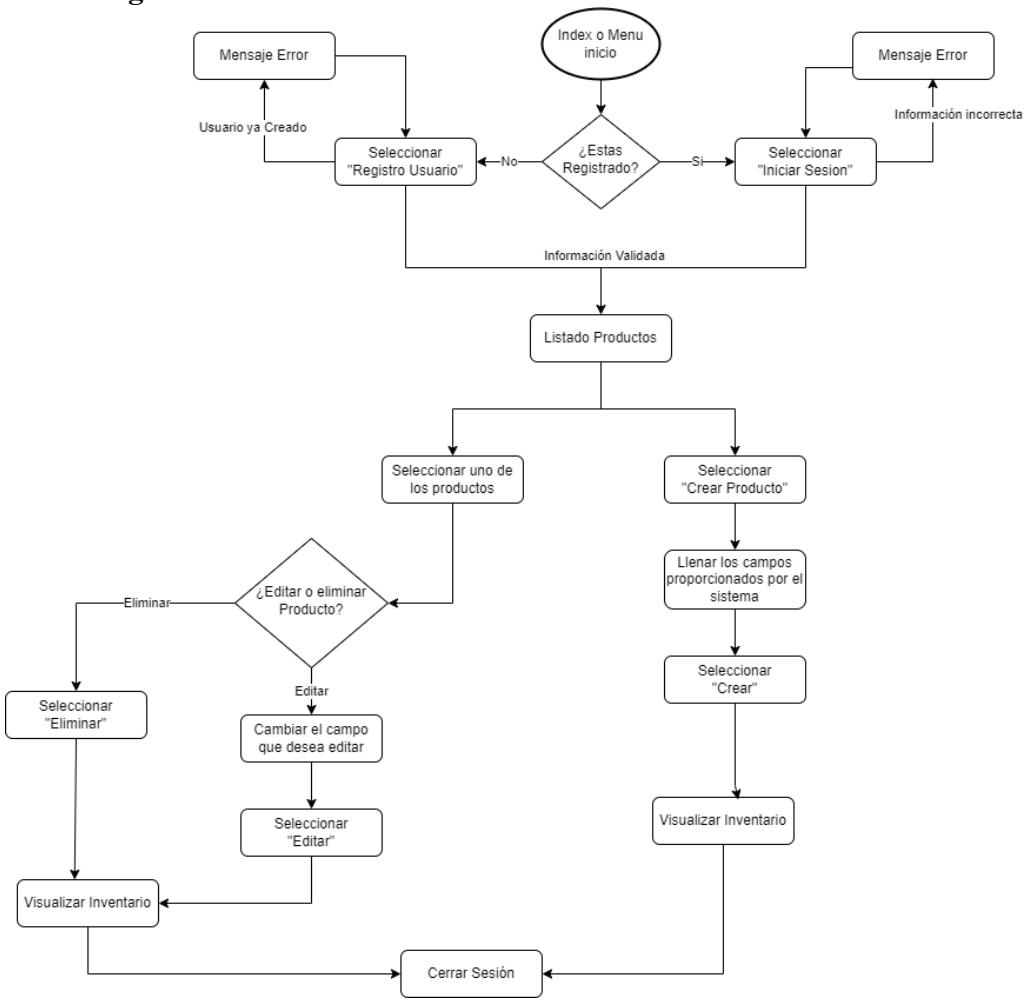


Imagen 5. *Diagrama funcional del sistema*

Fuente: *Elaboración propia*

El anterior diagrama describe los procesos que se realizan por parte del usuario en la página web, también describe los diferentes caminos que pueden tomar los usuarios para llegar a cierta actividad deseada.

5.3.3 Diagrama de arquitectura en la nube

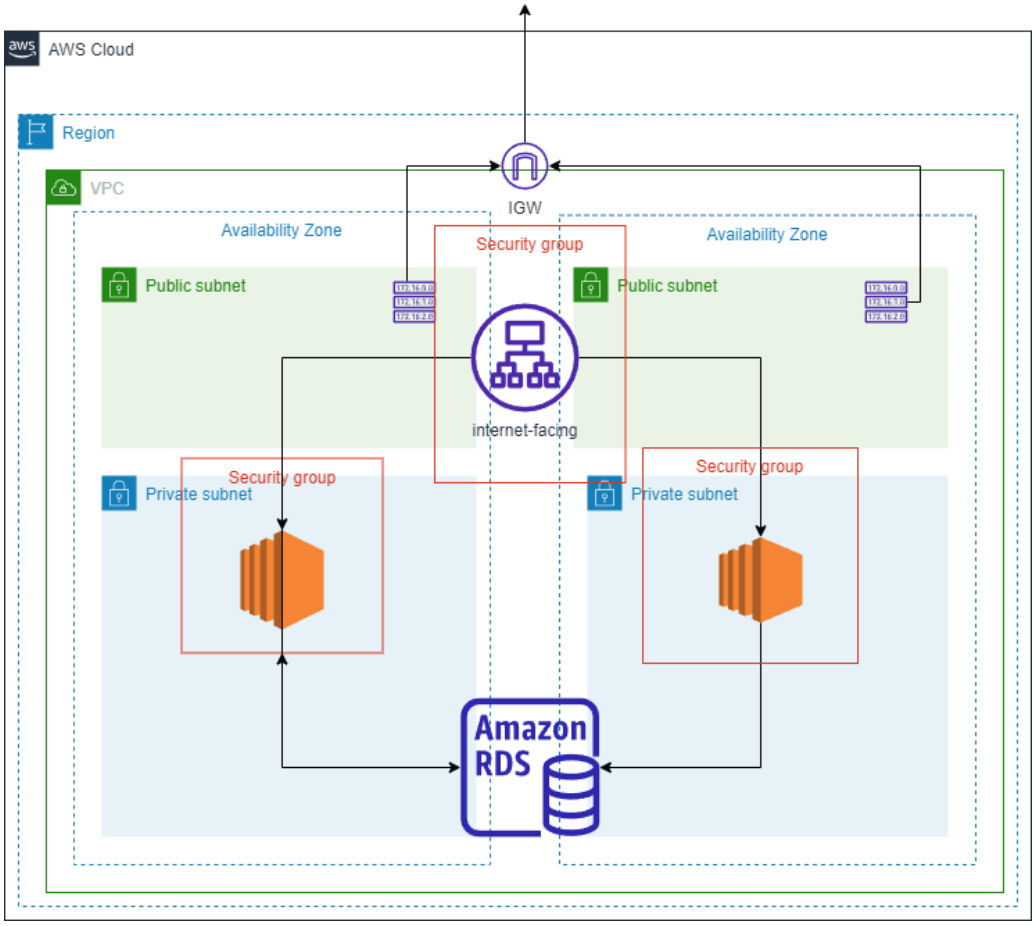
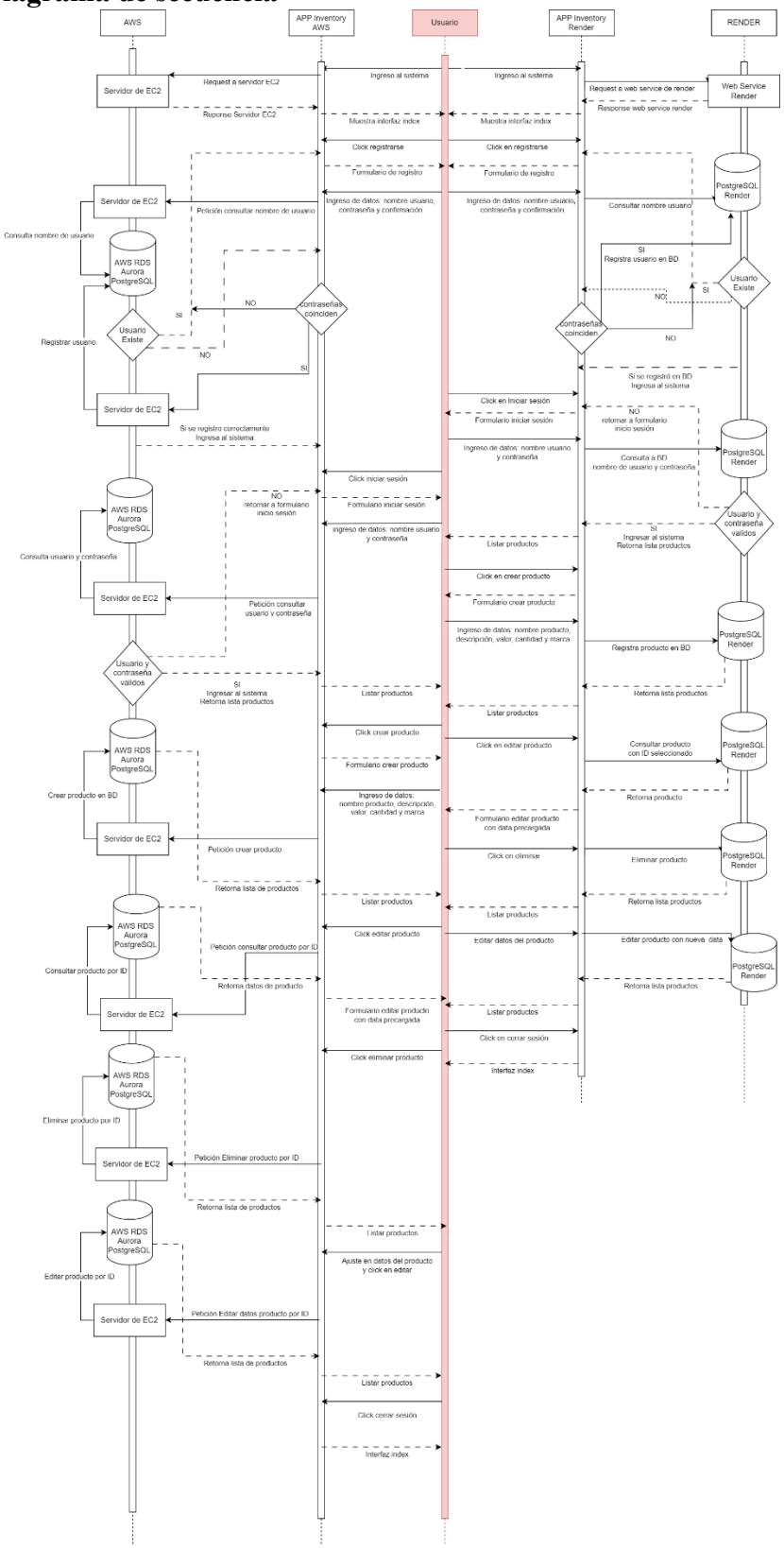


Imagen 6. *Diagrama de arquitectura*

Fuente: *Elaboración propia*

5.3.4 Diagrama de secuencia



Fuente: *Elaboración propia*

5.4.DESPLIEGUE

5.4.1 Backend Desplegado

Se utilizo como plataforma de servicios en la nube render.com, el cual con el código de GitHub mostrando anteriormente y la base de datos en PostgreSQL, nos permitió desplegarla y quedo con el siguiente enlace: https://inventario-9pqo.onrender.com/ y ec2-100-27-5-205.compute-1.amazonaws.com:8000/ para AWS.

Nota: Para poder ingresar a la url de AWS se deben encender las instancias ya que dadas las limitaciones permanecen apagadas en caso de querer visualizar el sistema en las url de render lo podrá hacer.

5.4.2 Rutas y funciones

Ruta AWS	Ruta Render	Función
http://ec2-100-27-5-205.compute-1.amazonaws.com:8000/	https://inventario-9pqo.onrender.com/	Página principal
http://ec2-100-27-5-205.compute-1.amazonaws.com:8000/RegistroU/	https://inventario-9pqo.onrender.com/RegistroU/	Registro de los usuarios
http://ec2-100-27-5-205.compute-1.amazonaws.com:8000/login/	https://inventario-9pqo.onrender.com/login/	Inicio Sesión
http://ec2-100-27-5-205.compute-1.amazonaws.com:8000/inventario/	https://inventario-9pqo.onrender.com/inventario/	Inventario principal del usuario
http://ec2-100-27-5-205.compute-1.amazonaws.com:8000/crearProducto/	https://inventario-9pqo.onrender.com/crearProducto/	Creación de los diferentes productos de la empresa
http://ec2-100-27-5-205.compute-1.amazonaws.com:8000/detalleProducto/1/	https://inventario-9pqo.onrender.com/detalleProducto/1/	Detalle de cada producto el cual contiene el id del producto

		para su edición o eliminación

Tabla 2. *Rutas con sus funciones RENDER*

Fuente: *Elaboración propia*

Estas son las diferentes rutas que contiene el sistema en cuanto frontend, ya que en el backend cuando se ingresa en el detalle del producto este tiene dos rutas a seguir mediante direcciones diferentes las cuales son eliminar y editar a diferencia del frontend que no tiene rutas ya que se hace de manera automática el cambio en el listado.

5.4.3 Limitaciones de los proveedores de nube

El backend fue desarrollado mediante Django el cual es un framework de desarrollo web por parte de Python y el frontend se realizó mediante HTML y Boostrap las cuales son herramientas para el diseño.

En complemento se utilizó el IDE de visual studio code para su codificación, luego mediante GitHub se realizó el repositorio para finalmente mediante render.com poder hacer el despliegue de la aplicación.

Al utilizar render se usó un plan gratuito ya que el proyecto se realizó sin fines lucrativos este tiene ciertas limitaciones como lo son:

1. Servicio Web:
    - 750 horas activa la página web por mes, estas horas se reinician cada primero de mes
    - La cuenta tiene 100 GB de salida de ancho de banda para cada servicio utilizado
    - La página se cierra automáticamente después de estar 15 minutos inactiva, esto causa cierto retraso a la hora de utilizarlo la primera vez

2. PostgreSQL base de datos:
    - Después de 90 días la base de datos se suspende
    - 14 días de tiempo extra para así poder renovar membresía o pagar una
    - Solo se puede tener una base de datos activa a la vez

Al utilizar la capa gratuita de AWS se cuenta con las siguientes restricciones y limitaciones:

1. AWS EC2:
    - 45000 minutos de uso en instancias t2 y t3 micro al mes.
    - 1 año de uso gratuito.
    - Instancias de Linux, Windows, RHEL y SLES.
2. AWS RDS:
    - 45000 minutos de uso de instancias de base de datos db.t2.micro, db.t3.micro y db.t4g.micro al mes.
    - Únicamente habilitado para single-AZ o una sola zona de disponibilidad.
    - 20 GB de almacenamiento general (SSD).
    - 20 GB para almacenamiento de respaldos de la base de datos.
    - Solo para motores de bases de datos (MySQL, MariaDB, PostgreSQL).
3. AWS ELB:
    - 45000 minutos de uso compartido entre balanceadores clásicos y de aplicaciones.
    - 15 GB de procesamiento de datos para balanceadores clásicos.
    - 15 unidades de capacidad de balanceo para aplicaciones.

5.4.4 Set de pruebas y resultados

5.4.4.1 Set de pruebas

HU01 - Registro de Usuario:

CP001: Registro exitoso con caracteres alfanuméricos

Entrada: Nombre de usuario = "Usuario123", Contraseña = "Password456"

Resultado esperado: Redirección exitosa después del registro.

CP002: Registro con caracteres especiales en el nombre de usuario

Entrada: Nombre de usuario = "User@123", Contraseña = "Password456"

Resultado esperado: Mostrar un mensaje de error indicando que solo se permiten caracteres alfanuméricos.

CP003: Registro sin contraseña

Entrada: Nombre de usuario = "Usuario123", Contraseña = ""

Resultado esperado: Mostrar un mensaje de error indicando que la contraseña es obligatoria.

CP004: Interfaz de usuario intuitiva

Entrada: N/A (evaluación de la interfaz)

Resultado esperado: La interfaz debe ser fácil de entender y usar.

CP005: Color del botón de registro

Entrada: N/A (evaluación del botón de registro)

Resultado esperado: El botón de registro debe tener un color azul.

HU02 - Inicio de Sesión:

CP006: Inicio de sesión exitoso con nombre de usuario

Entrada: Nombre de usuario = "Usuario123", Contraseña = "Password456"

Resultado esperado: Redirección exitosa después del inicio de sesión.

CP007: Inicio de sesión exitoso con correo electrónico

Entrada: Correo electrónico = "usuario@example.com", Contraseña = "Password456"

Resultado esperado: Redirección exitosa después del inicio de sesión.

CP008: Inicio de sesión sin contraseña

Entrada: Nombre de usuario = "Usuario123", Contraseña = ""

Resultado esperado: Mostrar un mensaje de error indicando que la contraseña es obligatoria.

CP009: Interfaz de usuario intuitiva

Entrada: N/A (evaluación de la interfaz)

Resultado esperado: La interfaz debe ser fácil de entender y usar.

CP010: Color del botón de inicio de sesión

Entrada: N/A (evaluación del botón de inicio de sesión)

Resultado esperado: El botón de inicio de sesión debe tener un color azul.

HU03 - Registro de Productos:

CP011: Registro de producto exitoso

Entrada: Nombre del producto = "Producto1", Cantidad = 10, Valor = 50, Descripción = "Descripción del producto"

Resultado esperado: Se debe asignar automáticamente un ID al producto registrado.

CP012: Registro sin descripción

Entrada: Nombre del producto = "Producto1", Cantidad = 10, Valor = 50, Descripción = ""

Resultado esperado: El registro del producto debe ser exitoso, incluso sin una descripción.

CP013: Cantidad y valor no numéricos

Entrada: Nombre del producto = "Producto1", Cantidad = "Diez", Valor = "Cincuenta", Descripción = "Descripción del producto"

Resultado esperado: Mostrar mensajes de error indicando que la cantidad y el valor deben ser números.

CP014: Interfaz de registro de producto intuitiva

Entrada: N/A (evaluación de la interfaz)

Resultado esperado: La interfaz de registro de producto debe ser fácil de entender y usar.

CP015: Color del botón de registro de producto

Entrada: N/A (evaluación del botón de registro de producto)

Resultado esperado: El botón de registro de producto debe tener un color azul.

HU04 - Listado de Productos:

CP016: Listado de productos asociados al usuario

Entrada: Usuario = "Usuario123" (con productos asociados)

Resultado esperado: Mostrar una tabla con los productos del usuario, incluyendo nombre, cantidad, valor y descripción.

CP017: Listado de productos sin productos asociados

Entrada: Usuario = "Usuario456" (sin productos asociados)

Resultado esperado: Mostrar un mensaje indicando que no hay productos asociados a ese usuario.

CP018: Interfaz de listado de productos intuitiva

Entrada: N/A (evaluación de la interfaz)

Resultado esperado: La interfaz de listado de productos debe ser fácil de entender y usar.

CP019: Color del botón de edición del producto

Entrada: N/A (evaluación del botón de edición del producto)

Resultado esperado: El botón de edición del producto debe tener un color amarillo.

HU05 - Edición de Productos:

CP020: Precarga de datos en el formulario de edición

Entrada: Producto seleccionado para editar

Resultado esperado: El formulario de edición debe mostrar los datos del producto seleccionado para su edición.

CP021: Edición exitosa de producto

Entrada: Nombre del producto = "ProductoEditado", Cantidad = 15, Valor = 100, Descripción = "Descripción editada"

Resultado esperado: El producto debe actualizarse con los nuevos valores en la base de datos.

CP022: Edición con descripción vacía

Entrada: Nombre del producto = "ProductoEditado", Cantidad = 15, Valor = 100, Descripción = ""

Resultado esperado: El producto debe actualizarse con una descripción vacía en la base de datos.

CP023: Cantidad y valor no numéricos en la edición

Entrada: Nombre del producto = "ProductoEditado", Cantidad = "Quince", Valor = "Cien", Descripción = "Descripción editada"

Resultado esperado: Mostrar mensajes de error indicando que la cantidad y el valor deben ser números.

CP024: Interfaz de edición de producto intuitiva

Entrada: N/A (evaluación de la interfaz)

Resultado esperado: La interfaz de edición de producto debe ser fácil de entender y usar.

CP025: Color del botón de edición de producto

Entrada: N/A (evaluación del botón de edición de producto)

Resultado esperado: El botón de edición de producto debe tener un color amarillo.

CP026: Color del botón de eliminación de producto

Entrada: N/A (evaluación del botón de eliminación de producto)

Resultado esperado: El botón de eliminación de producto debe tener un color rojo.

HU06 - Eliminación de Productos:

CP027: Eliminación exitosa de producto

Entrada: Producto seleccionado para eliminar

Resultado esperado: El producto debe eliminarse de la base de datos.

CP028: Interfaz de eliminación de producto intuitiva

Entrada: N/A (evaluación de la interfaz)

Resultado esperado: La interfaz de eliminación de producto debe ser fácil de entender y usar.

CP029: Color del botón de edición de producto

Entrada: N/A (evaluación del botón de edición de producto)

Resultado esperado: El botón de edición de producto debe tener un color amarillo.

CP030: Color del botón de eliminación de producto

Entrada: N/A (evaluación del botón de eliminación de producto)

Resultado esperado: El botón de eliminación de producto debe tener un color rojo.

HU07 - Cierre de Sesión:

CP031: Cierre de sesión exitoso

Entrada: N/A (hacer clic en el botón de cierre de sesión)

Resultado esperado: Redirección a la página de inicio de sesión.

CP032: Interfaz de cierre de sesión intuitiva

Entrada: N/A (evaluación de la interfaz)

Resultado esperado: La interfaz de cierre de sesión debe ser fácil de entender y usar.

HU08 - Acceso al Sistema:

CP033: Acceso al sistema disponible

Entrada: N/A (intentar acceder al sistema)

Resultado esperado: El sistema debe estar disponible y accesible a través del proveedor de nube.

CP034: Escalabilidad del sistema

Entrada: N/A (evaluación del desempeño)

Resultado esperado: El sistema desplegado debe mostrar un rendimiento aceptable bajo carga.

CP035: Capa gratuita del proveedor de nube

Entrada: N/A (evaluación del plan de la nube)

Resultado esperado: El proveedor de nube debe ofrecer una capa gratuita para optimizar recursos.

CP036: Acceso al sistema desde diferentes dispositivos

Entrada: N/A (intentar acceder desde diferentes dispositivos)

Resultado esperado: El sistema debe ser accesible desde dispositivos como celular, computador, tablet, etc.

CP037: Servicios necesarios para la implementación

Entrada: N/A (evaluación de los servicios del proveedor de nube)

Resultado esperado: El proveedor de nube debe contar con los servicios necesarios para la implementación del sistema.

5.4.4.2 Resultados

ID Caso de Prueba	Estado de certificación	Estado CP
CP001	Certificado	Satisfactorio
CP002	Certificado	Satisfactorio
CP003	Certificado	Satisfactorio
CP004	Certificado	Satisfactorio
CP005	Certificado	Satisfactorio
CP006	Certificado	Satisfactorio
CP007	Certificado	Satisfactorio
CP008	Certificado	Satisfactorio
CP009	Certificado	Satisfactorio
CP010	Certificado	Satisfactorio
CP011	Certificado	Satisfactorio
CP012	Certificado	Satisfactorio
CP013	Certificado	Satisfactorio
CP014	Certificado	Satisfactorio
CP015	Certificado	Satisfactorio
CP016	Certificado	Satisfactorio
CP017	Certificado	Satisfactorio
CP018	Certificado	Satisfactorio
CP019	Certificado	Satisfactorio
CP020	Certificado	Satisfactorio
CP021	Certificado	Satisfactorio
CP022	Certificado	Satisfactorio
CP023	Certificado	Satisfactorio
CP024	Certificado	Satisfactorio
CP025	Certificado	Satisfactorio
CP026	Certificado	Satisfactorio
CP027	Certificado	Satisfactorio
CP028	Certificado	Satisfactorio
CP029	Certificado	Satisfactorio
CP030	Certificado	Satisfactorio
CP031	Certificado	Satisfactorio
CP032	Certificado	Satisfactorio
CP033	Certificado	Satisfactorio
CP034	Certificado	Satisfactorio
CP035	Certificado	Satisfactorio
CP036	Certificado	Satisfactorio
CP037	Certificado	Satisfactorio

Fuente: *Elaboración propia*

5.4.1 Pruebas funcionales

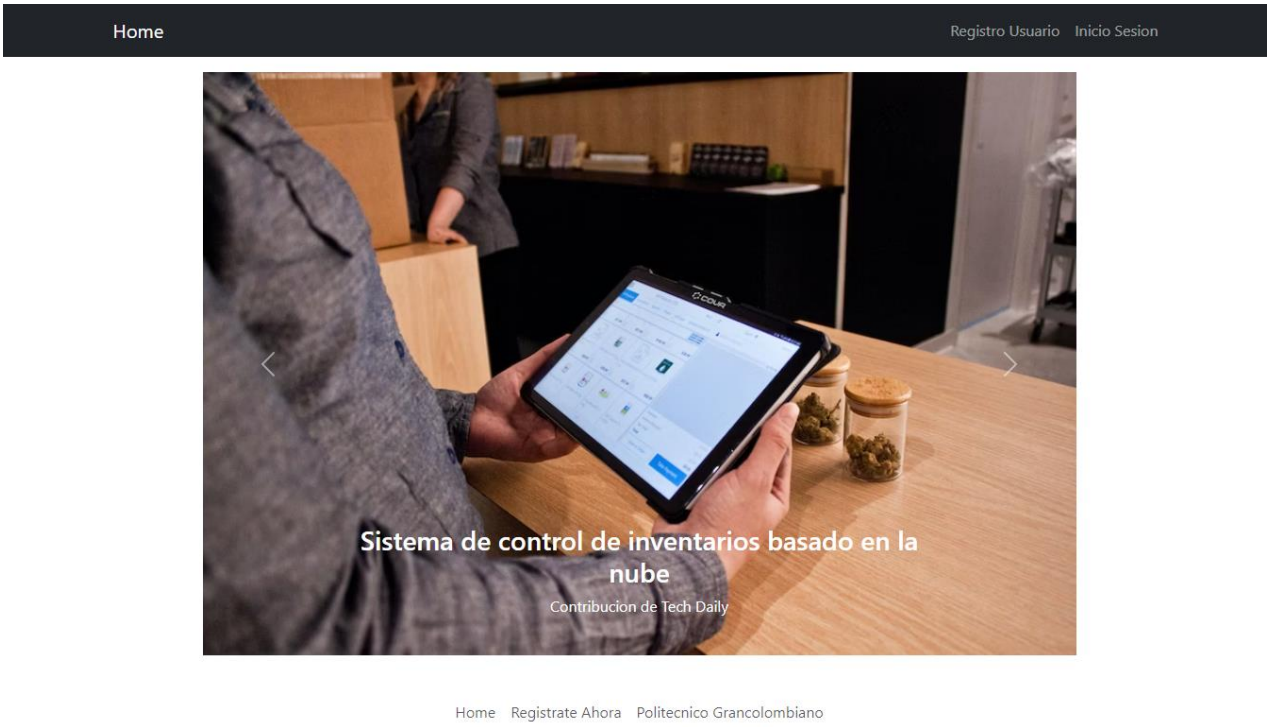


Imagen 7. *Página principal*

Fuente: *Elaboración propia*

Esta página será la principal a la que entre el usuario apenas ingrese a la plataforma, tiene un carrusel el cual contiene algunas de los objetivos de porque se diseñó la plataforma, esta vista tiene como url la siguiente: https://inventario-9pqo.onrender.com/. La barra de navegación nos permitirá el entrar a dos apartados diferentes los cuales son el inicio de sesión y el registro de los usuarios.

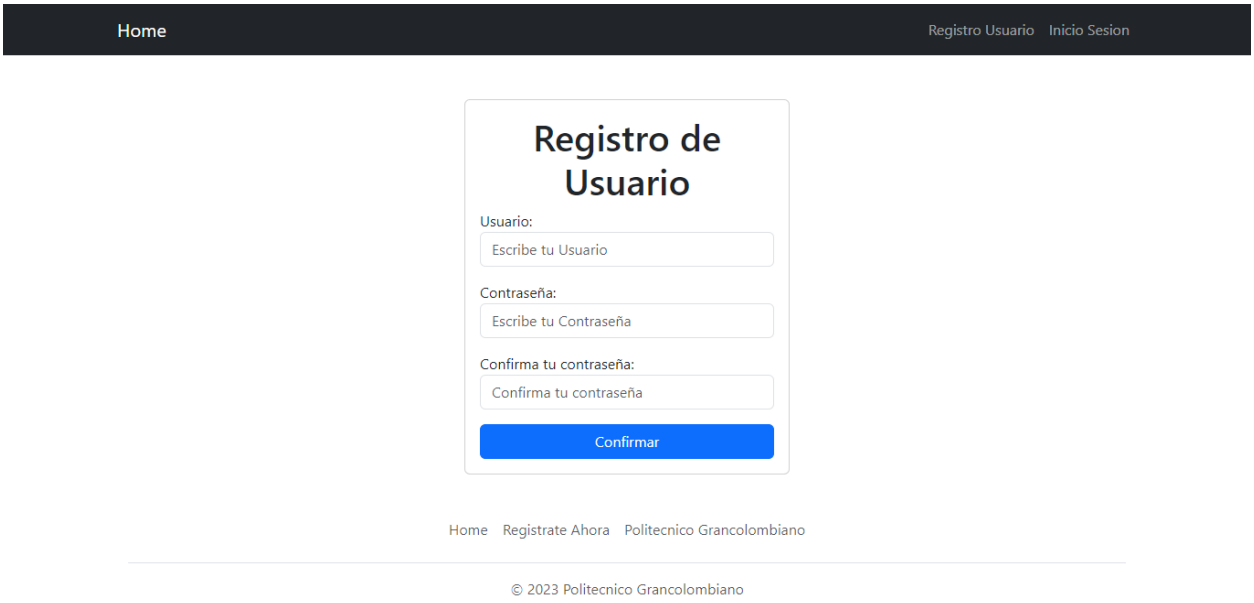


Imagen 8. *Registro Usuario*

Fuente: *Elaboración propia*

En esta vista podremos observar el registro de usuario que contiene los campos de usuario, contraseña y confirmar contraseña. Tiene como url la siguiente: https://inventario-9pqo.onrender.com/RegistroU/. Para realizar una prueba probaremos primero con una contraseña y otra en la confirmación, también probaremos con un usuario ya creado, esto con el fin de ver que genera ciertos errores.

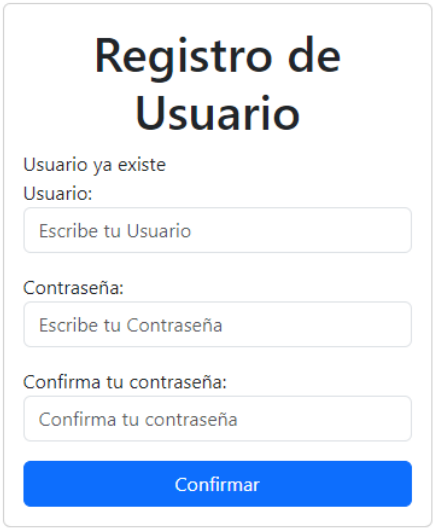


Imagen 9. *Error Usuario ya existe*

Fuente: *Elaboración propia*

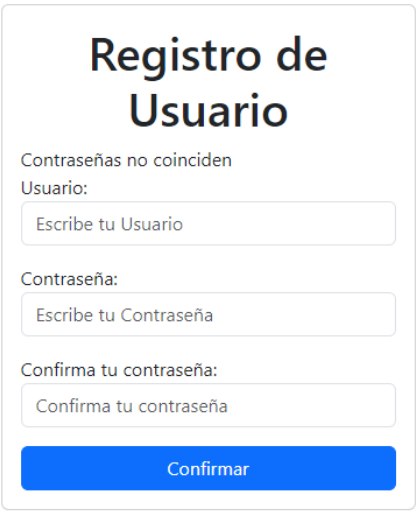


Imagen 10. *Error Contraseña no coinciden*

Fuente: *Elaboración propia*

Y ahora se realizará la prueba con un usuario no creado y con contraseñas iguales.

Usuario: Poli – Contraseña: Poli123

Imagen 11. *Inventario*

Fuente: *Elaboración propia*

Y como vemos entro a la plataforma satisfactoriamente, esta vista tiene como url la siguiente: https://inventario-9pqo.onrender.com/inventario/. Para hacer pruebas en este apartado podemos crear un producto y así revisar si quedase con el usuario Poli.

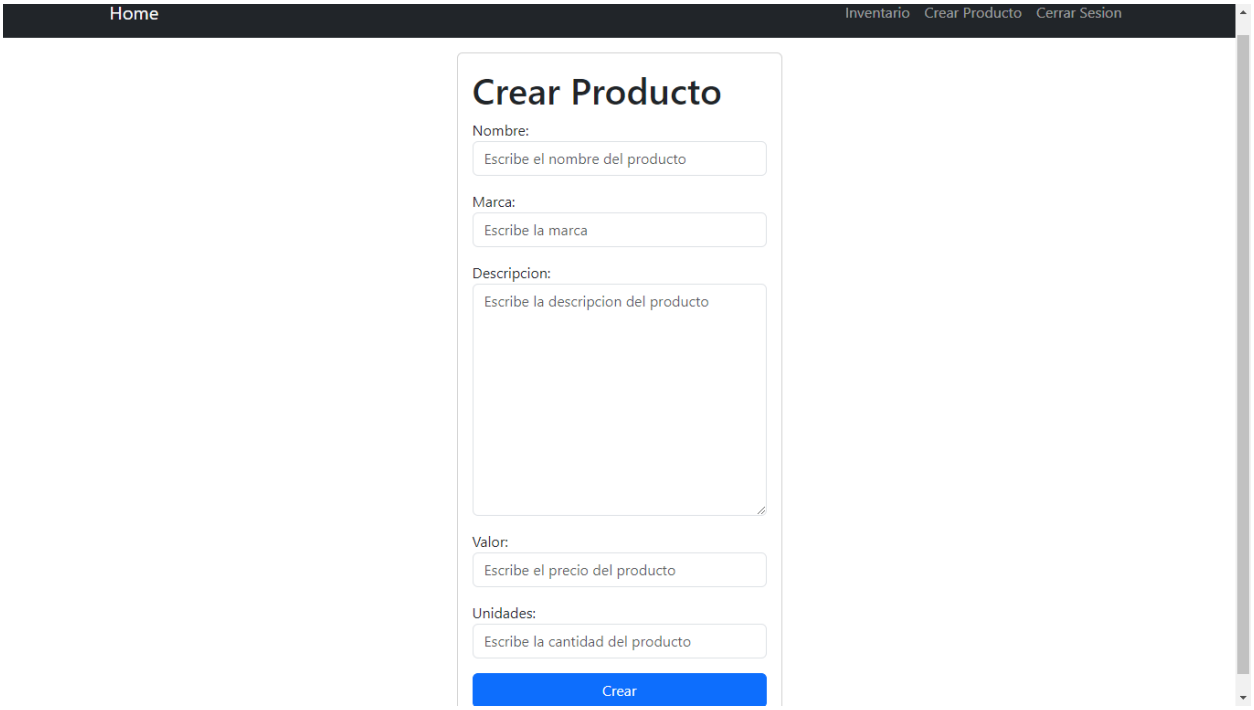


Imagen 12. *Crear Producto*

Fuente: *Elaboración propia*

Aquí se pondrán unos datos de un producto cualquiera para si verificar si todo funciona correctamente. Esta vista tiene como url la siguiente: https://inventario-9pqo.onrender.com/crearProducto/

Imagen 13. *Crear Producto*

Fuente: *Elaboración propia*

Como se observa los datos son creado en el inventario con el usuario Poli el cual fue creado con anterioridad, ahora por otro lado también tenemos el botón de editar el cual nos lleva a un apartado para eliminar el producto o editarlo.

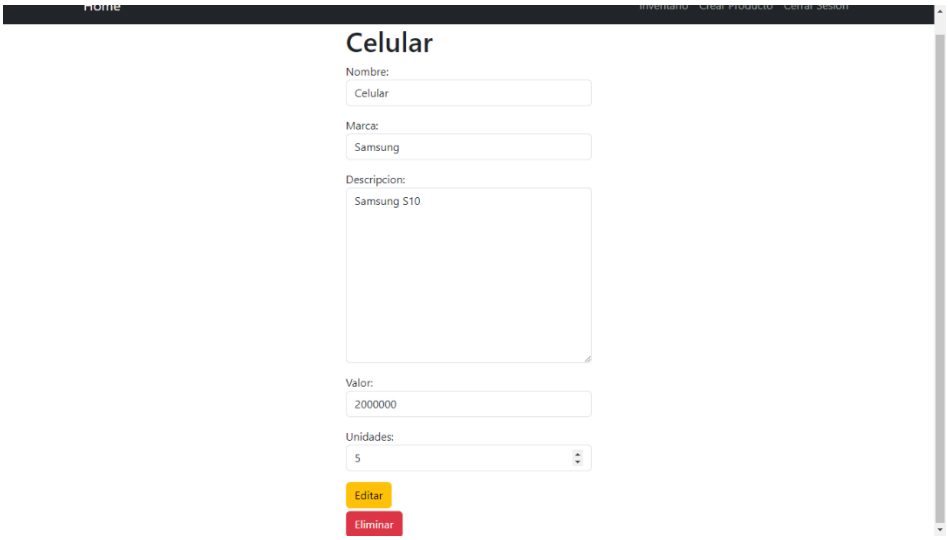


Imagen 14. *Detalle Producto*

Fuente: *Elaboración propia*

En esta vista vamos a realizar algún cambio a los datos para así verificar su edición y por último eliminarlo. Esta vista tiene como url la siguiente: https://inventario-9pqo.onrender.com/detalleProducto/4/

Imagen 15. *Detalle Producto*

Fuente: *Elaboración propia*

Como se observa hay cambios en los apartados ahora pasaremos a la eliminación:

Imagen 16. *Inventario*

Fuente: *Elaboración propia*

En el inventario ya quedaría vacío después de la eliminación de su único objeto.

Hay otra vista en la página principal la cual es el inicio de sesión, se hará la prueba con un usuario no existente y otra con el Poli que acabamos de crear.

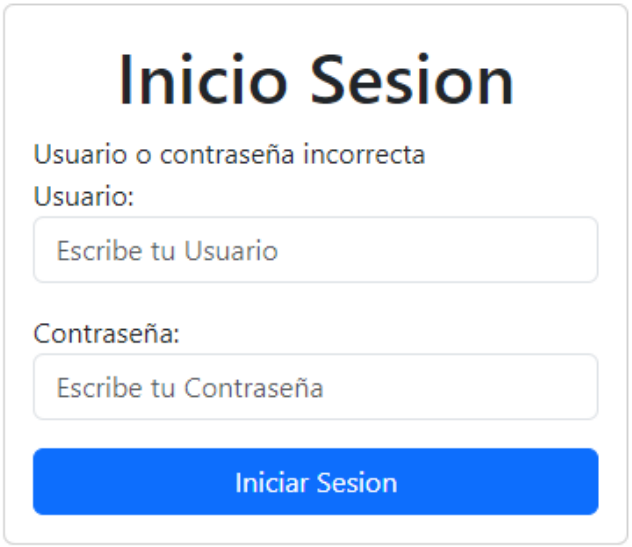


Imagen 17. *Error Inicio de sesión*

Fuente: *Elaboración propia*

Como vemos si probamos con un usuario o contraseña erróneos nos da un mensaje de error, en cambio sí ponemos el usuario del Poli nos debería pasar al inventario vacío.

Imagen 18. *Inventario Usuario Poli*

Fuente: *Elaboración propia*

Por último, se tienen dos Footers además de la barra de navegación que son las siguientes:

Imagen 19. *Footer Inventario*

Fuente: *Elaboración propia*

Este footer o pie de página contiene un home que direcciona a la página principal, un crear producto para la creación de los productos y “Politécnico Grancolombiano” que dirección a la página poli.edu.co

Imagen 20. *Footer apartados principales*

Fuente: *Elaboración propia*

Este footer o pie de página contiene un home que direcciona a la página principal, un Registrarse ahora para la vista de registro y “Politécnico Grancolombiano” que dirección a la página poli.edu.co

5.4.2 Despliegue en AWS

1. Inicialmente se tuvo que crear una cuenta en AWS para que de esta manera se puedan utilizar los servicios de la capa gratuita de AWS, este proceso consta de 5 simples y sencillos pasos:
    a. Agregar y verificar correo electrónico, definir el nombre de la cuenta y crear una contraseña para el usuario raíz.

Imagen 21. *Registro de AWS*

Fuente: *AWS*

b. Especificar para que fines se va a utilizar los servicios de AWS y agregar información de contacto.

Imagen 22. *Registro de AWS*

Fuente: *AWS*

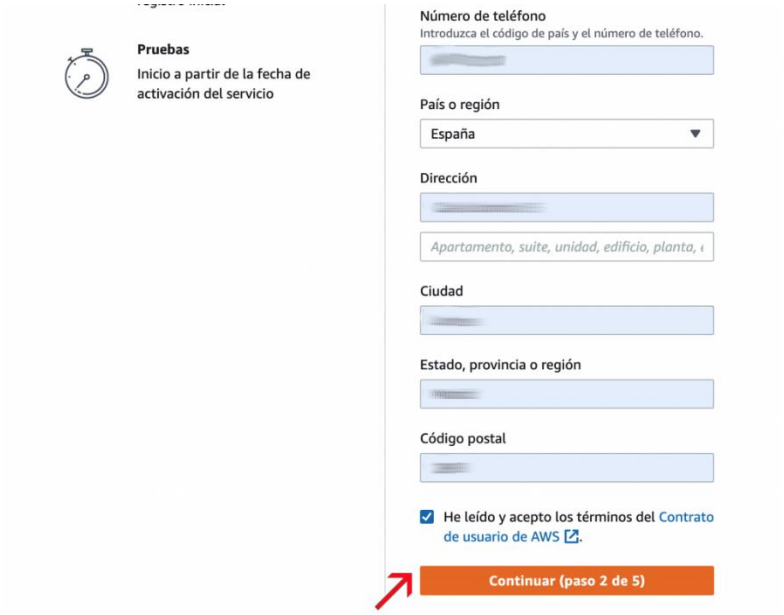


Imagen 23. *Registro de AWS*

Fuente: *AWS*

c. Especificar información del método de pago, dado que es obligatorio se debe contar con una tarjeta de crédito o débito, donde en caso de superar los límites de uso que ofrece la capa gratuita de AWS se generarán los cobros generados.

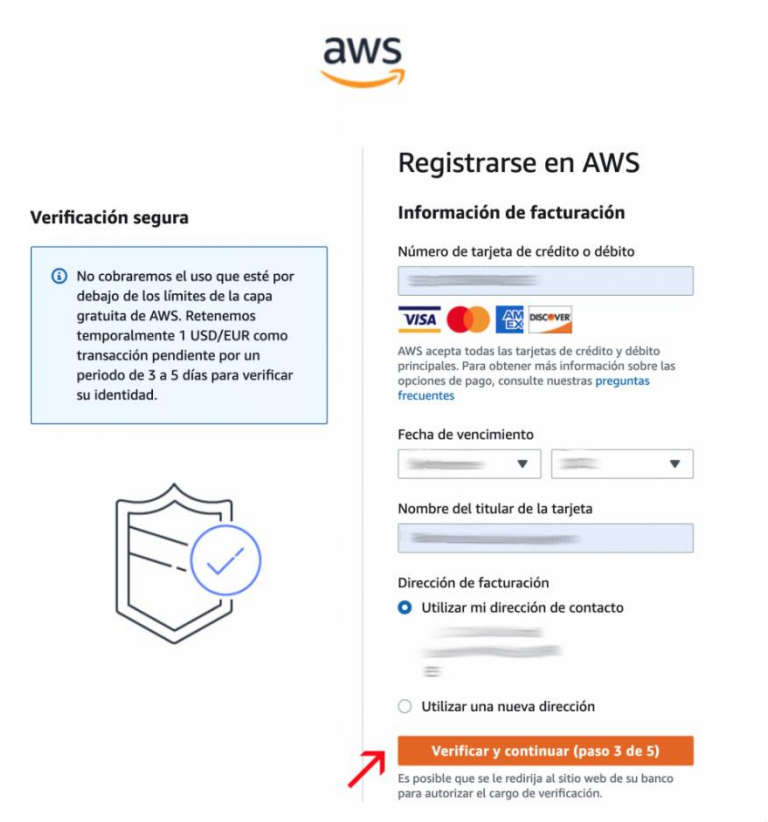


Imagen 24. *Registro de AWS*

Fuente: *AWS*

d. Verificar de identidad con un código enviado al número de teléfono especificado

Imagen 25. *Registro de AWS*
Fuente: *AWS*

e. Seleccionar plan de soporte

Imagen 26. *Plan soporte de AWS*
Fuente: *AWS*

2. Con la cuenta ya creada, lo primero que se tuvo que hacer fue utilizar el servicio de AWS IAM para crear los grupos, roles, y usuarios con los permisos necesarios para poder realizar el proceso de despliegue.

Se creó inicialmente los usuarios para lo cual fue necesario completar los siguientes cuatro pasos:

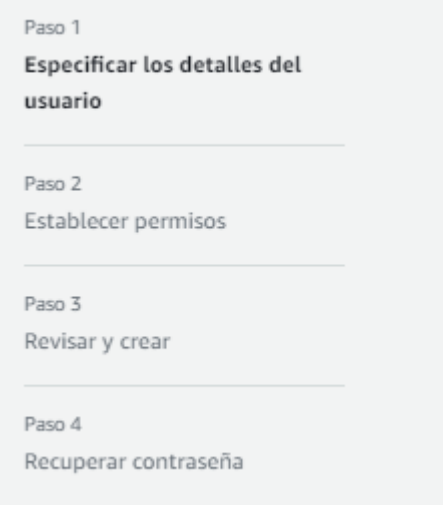


Imagen 27. *Creación de usuario IAM*
Fuente: *AWS*

a. Especificar los detalles del usuario:

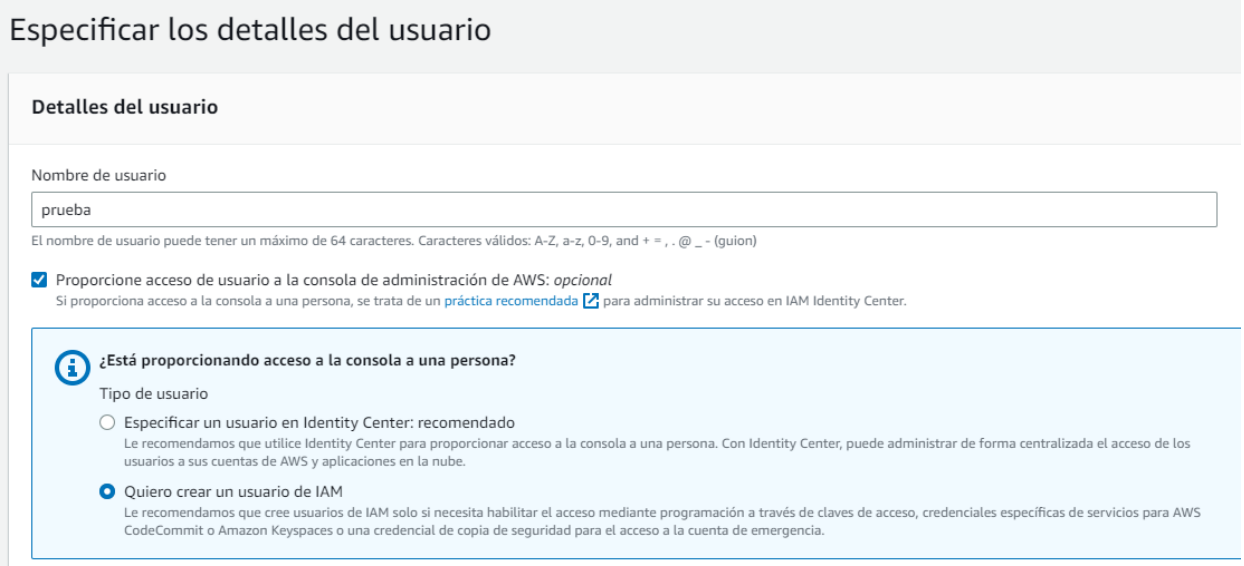


Imagen 28. *Detalles del usuario*
Fuente: *AWS*

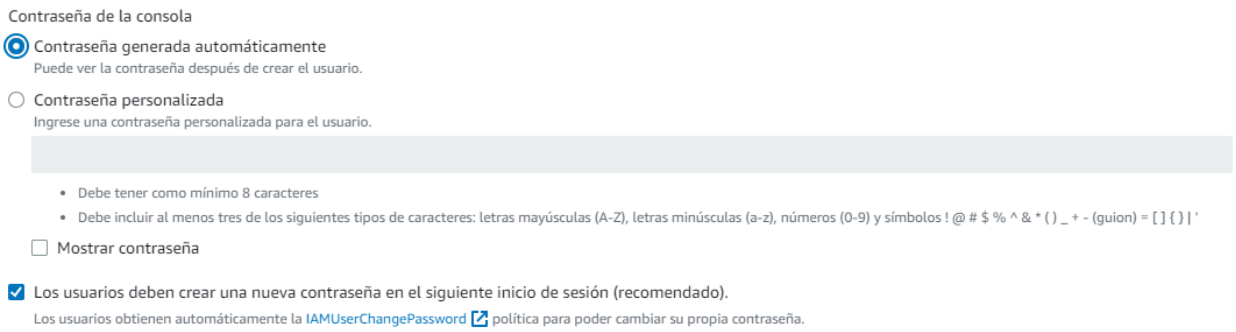


Imagen 29. *Detalles del usuario*
Fuente: *AWS*

b. Establecer permisos:

Estos permisos se pueden establecer de tres diferentes maneras:

- Agregando ese usuario a un grupo
- Copiando los permisos de otros usuarios ya creados
- Agregando directamente las políticas.

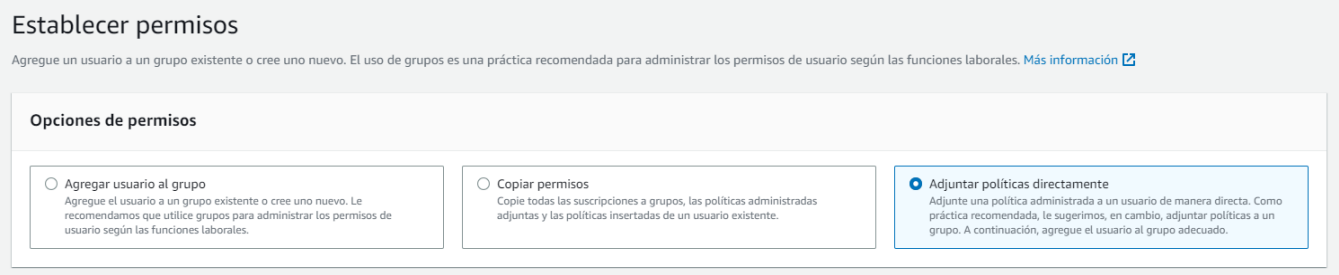


Imagen 30. *Permisos*
Fuente: *AWS*

c. Revisar información y crear usuario:

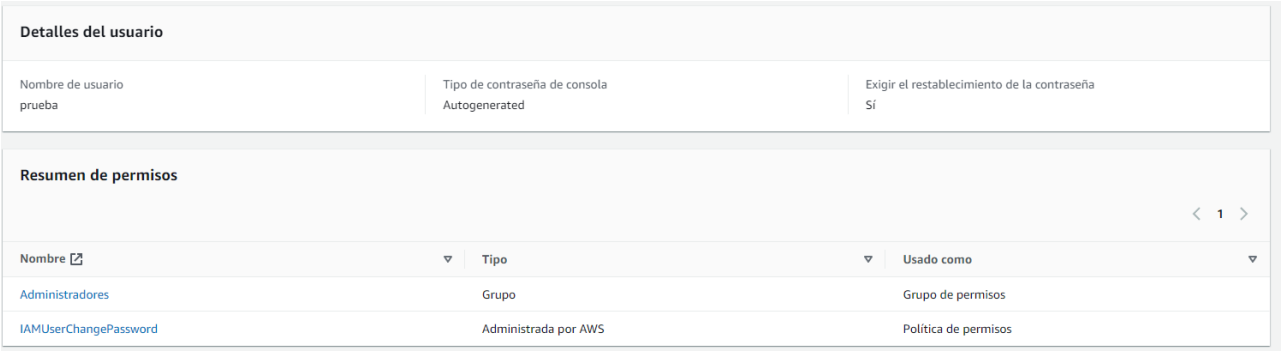


Imagen 31. *Información Usuario*
Fuente: *AWS*

d. Recuperar contraseña:

En este paso lo que haremos es descargar la contraseña para compartirla con la persona que hará uso del usuario creado.

3. Luego de crear las cuentas de IAM necesarias para los actores en el proceso de despliegue empezaremos creando una VPC (Virtual Private Cloud) donde alojaremos cada una de las instancias y servicios que se van a utilizar.

Imagen 32. *Información Usuario*
Fuente: *AWS*

En esta VPC creada asociaremos las subredes públicas y privadas las cuáles serán las que alojen los servicios utilizados según las necesidades.

En el despliegue realizado se ha utilizado una sola VPC, 6 subredes, 1 tabla de enrutamiento y 1 internet gateway.

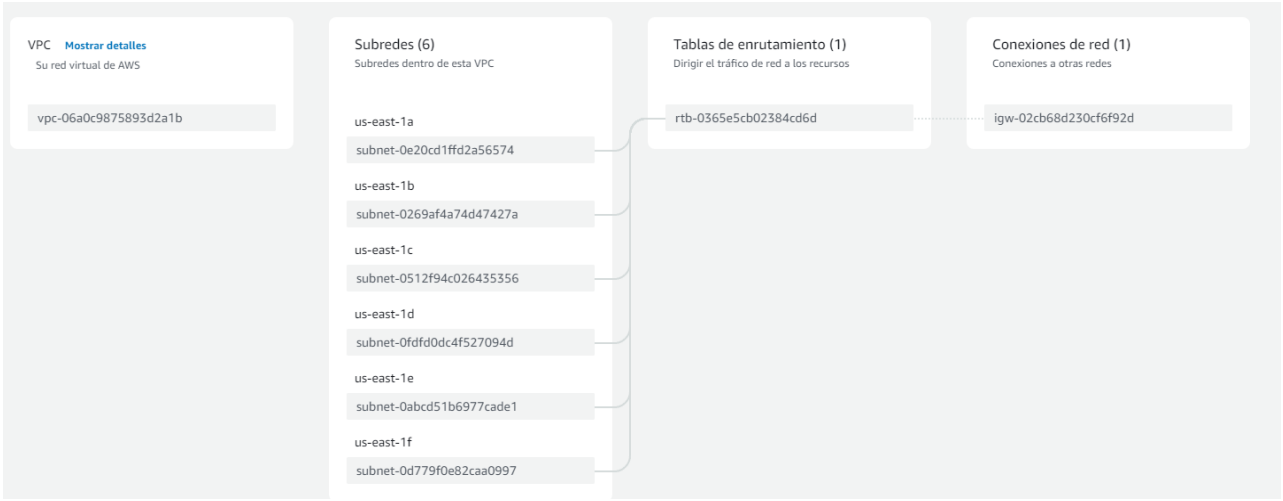


Imagen 33. *VPC*
Fuente: *AWS*

Como se puede visualizar en la imagen anterior cada subred está asociada a la misma región, pero a diferentes zonas de disponibilidad (us-east-1a, us-east-1b, us-east-1c, us-east-1d, us-east-1e, us-east-1f) y estas 6 subredes están asociadas a una misma tabla de enrutamiento la cual tiene una conexión a internet mediante el internet gateway.

ID de subred	Estado	VPC	CIDR IPv4	Direcciones IPv4 disponibles	Zona de disponibilidad	Tabla de enrutamiento
subnet-0512f94c026435356	Available	vpc-06a0c9875893d2a1b	172.31.0.0/20	4090	us-east-1c	rtb-0365e5cb02384cd6d
subnet-0abcd51b6977cade1	Available	vpc-06a0c9875893d2a1b	172.31.48.0/20	4090	us-east-1e	rtb-0365e5cb02384cd6d
subnet-0d779f0e82caa0997	Available	vpc-06a0c9875893d2a1b	172.31.64.0/20	4091	us-east-1f	rtb-0365e5cb02384cd6d
subnet-0fdfd0dc4f527094d	Available	vpc-06a0c9875893d2a1b	172.31.80.0/20	4091	us-east-1d	rtb-0365e5cb02384cd6d
subnet-0269af4a74d47427a	Available	vpc-06a0c9875893d2a1b	172.31.32.0/20	4091	us-east-1b	rtb-0365e5cb02384cd6d
subnet-0e20cd1ffd2a56574	Available	vpc-06a0c9875893d2a1b	172.31.16.0/20	4091	us-east-1a	rtb-0365e5cb02384cd6d

Imagen 34. *Subredes*
Fuente: *AWS*

De las 6 subredes asociadas a la tabla de enrutamiento solo 2 de estas son asociaciones explicitas y las otras 4 no lo son.

Asociaciones de subredes explícitas (2)

Editar asociaciones de subredes

Buscar asociación de subredes

Name	ID de subred	CIDR IPv4	CIDR IPv6
–	subnet-0abcd51b6977cade1	172.31.48.0/20	–
–	subnet-0d779f0e82caa0997	172.31.64.0/20	–

Subredes sin asociaciones explícitas (4)

Editar asociaciones de subredes

Las siguientes subredes no se han asociado explícitamente con ninguna tabla de enrutamiento y, por lo tanto, están asociadas a la tabla de enrutamiento principal:

Buscar asociación de subredes

Name	ID de subred	CIDR IPv4	CIDR IPv6
–	subnet-0512f94c026435356	172.31.0.0/20	–
–	subnet-0fdfd0dc4f527094d	172.31.80.0/20	–
–	subnet-0269af4a74d47427a	172.31.32.0/20	–
–	subnet-0e20cd1ffd2a56574	172.31.16.0/20	–

Imagen 35. *Asociación Subredes VPC*
Fuente: *AWS*

Una vez creada nuestra configuración de red empezaremos a realizar uso de los servicios de AWS para montar nuestra aplicación en la nube.

4. Inicialmente se creó asociado a la VPC y a una de las subredes una instancia haciendo uso de AWS EC2 la instancia creada es de tipo t2.micro la cual es una de las que acepta la capa gratuita de AWS, con sistema operativo Ubuntu 20.04 y unas características de capacidad básicas:
    - 1 CPU virtual (vCPU)
    - 8 gb SSD de uso general (gp3).
    - Asignación de IPs de manera automática

Luego se asigna un rol el cual nos permita utilizar la maquina desde el servicio de AWS System Manager el que nos provee una conexión directa a la instancia creada y la podemos acceder por comandos de consola y esta será nuestra herramienta principal para conectarnos y administrar esta instancia.

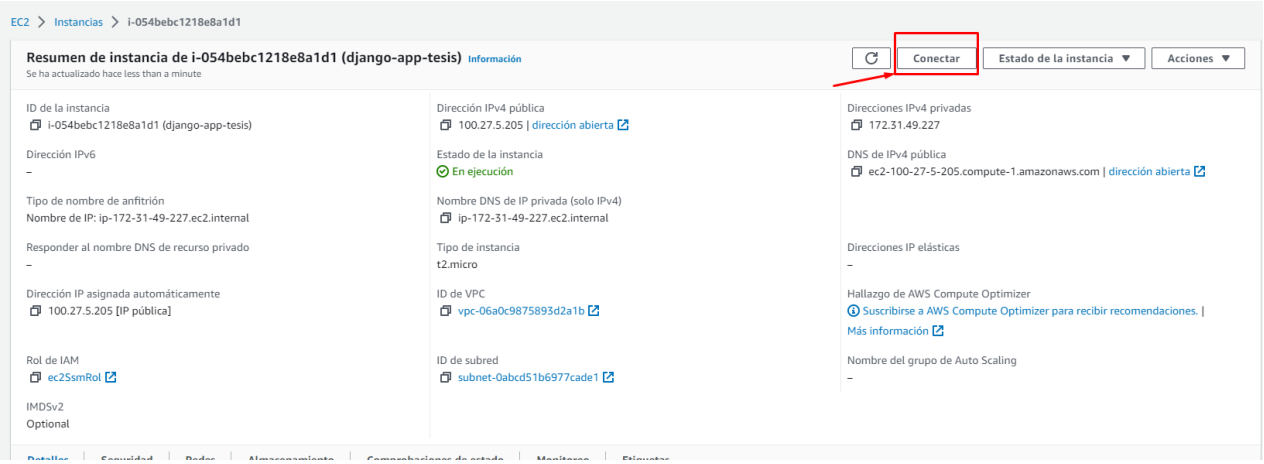


Imagen 36. *EC2 instancias*
Fuente: *AWS*

Al dar en conectar debemos seleccionar el administrador de sesiones y damos clic en conectar, al hacer esto se abrirá una pestaña nueva con la consola desde donde vamos a administrar nuestra instancia.

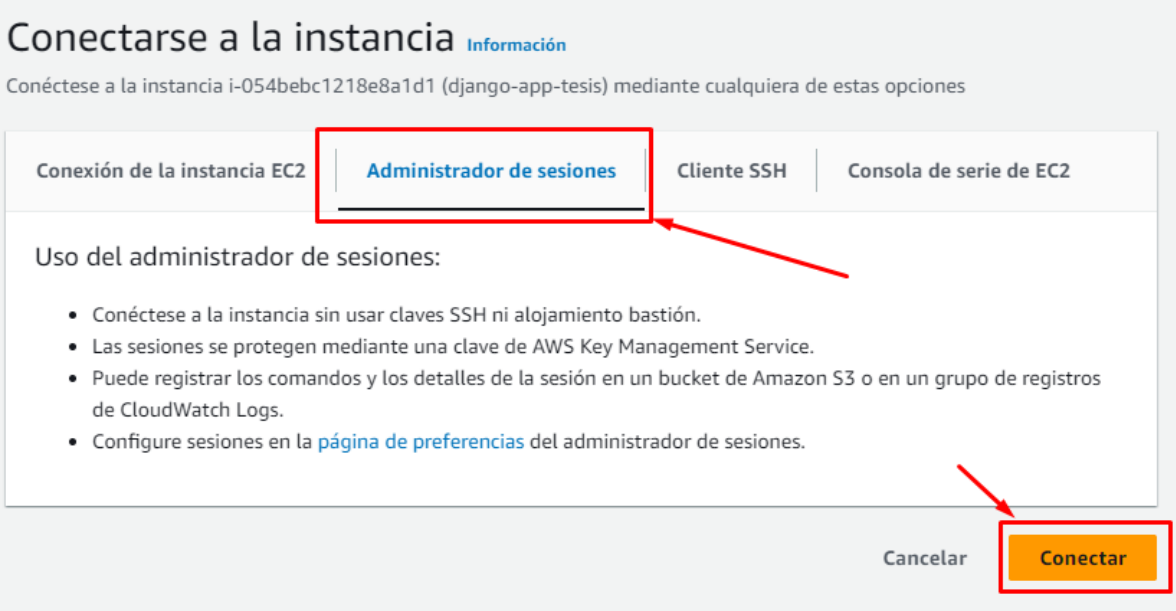


Imagen 37. *Conexión Instancia*
Fuente: *AWS*

En la consola digitaremos el comando bash para usar la vista de la consola en bash.

Imagen 38. *Comando bash*
Fuente: *AWS*

5. Para configurar el entorno de nuestra aplicación necesitamos instalar:

   - Docker
   - Docker-compose
   - Python
   - Pip

   Nota: En pruebas iniciales se instaló postgreSQL para levantar en una imagen de docker con la base de datos y otro con el proyecto de Django, pero en la implementación final se tiene una instancia de base de datos en el servicio de

AWS Relational Database Service (RDS) quien atiende las peticiones a la base de datos.

6. Al instalar y configurar nuestro entorno de ejecución de la aplicación, con la ayuda de git vamos a clonar nuestro repositorio en la instancia con el uso del comando básico de git clone podremos tener el código base de nuestra aplicación en nuestra instancia.
7. Una vez creada la instancia de EC2 podemos crear nuestra instancia de base de datos haciendo uso del servicio de AWS RDS.

   Nos vamos a ayudar con Aurora una opción proporcionada por AWS para la base de datos basada en serverless V2:

   Configuración: motor de base de datos, versiones, tipo de capacidad y tipo de red se especifican en este apartado.

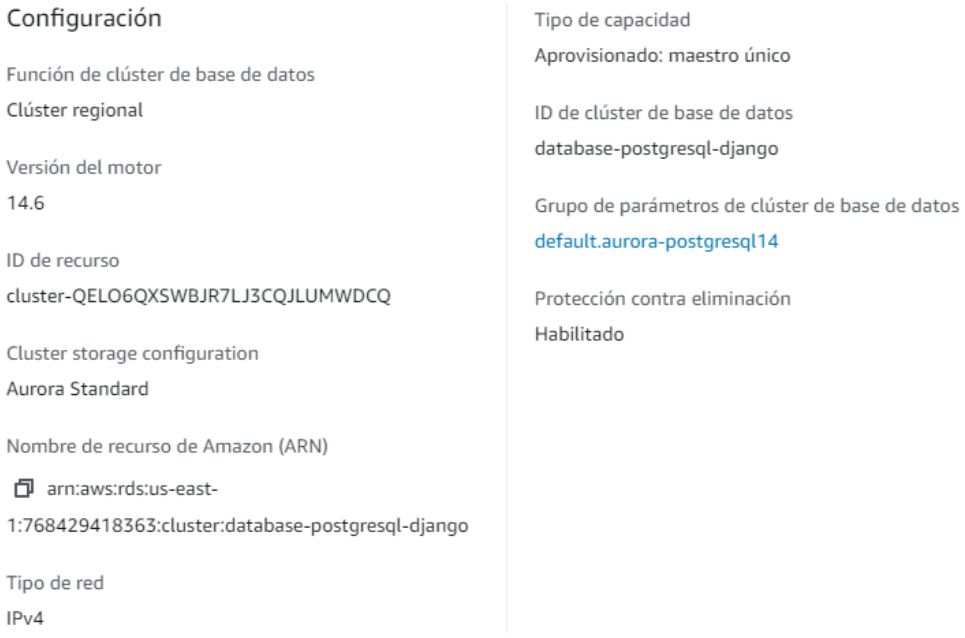


Imagen 39. *Configuración base de datos*
Fuente: *AWS*

Autenticación: manera de conectar y credenciales de conexión

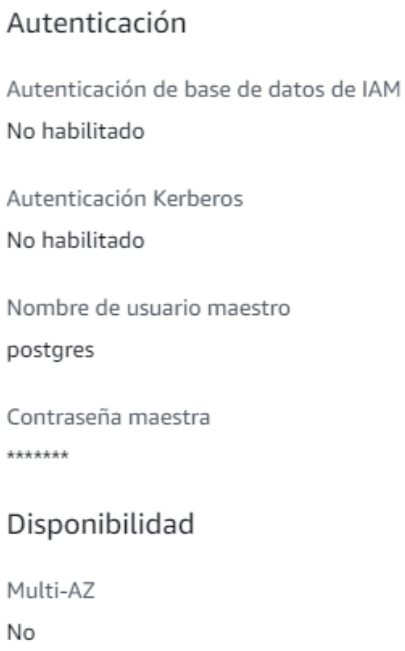


Imagen 40. *Autenticación base de datos*
Fuente: *AWS*

Cifrado: el cifrado es utilizado bajo un par de llaves almacenadas en el servicio de KMS.

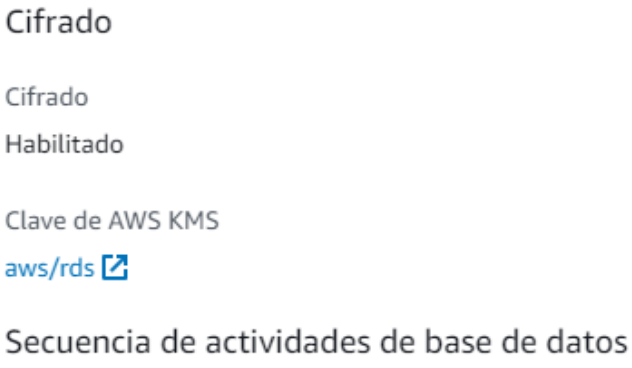
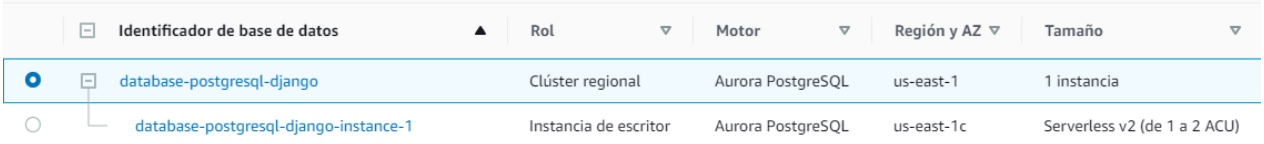


Imagen 41. *Cifrado base de datos*
Fuente: *AWS*

Al configurar nuestra base de datos con motor Aurora PostgreSQL nuestro resultado será un clúster en la región que se haya especificado y adentro una instancia para nuestra base de datos:



Identificador de base de datos	Rol	Motor	Región y AZ	Tamaño
database-postgresql-django	Clúster regional	Aurora PostgreSQL	us-east-1	1 instancia
database-postgresql-django-instance-1	Instancia de escritor	Aurora PostgreSQL	us-east-1c	Serverless v2 (de 1 a 2 ACU)

Imagen 42. *bases de datos*
Fuente: *AWS*

8. En nuestro proyecto deben estar un archivo de Docker el cual se encarga de ejecutar nuestra aplicación en una imagen de Docker, instalar las dependencias del proyecto las cuales se encuentran en el archivo requirements.txt y un archivo para el Docker compose ya que este es el que realiza las migraciones necesarias para la base de datos y conecta al servicio de RDS de AWS donde tenemos nuestra instancia de base de datos PostgreSQL.

Archivo Docker:

```
# syntax=docker/dockerfile:1
FROM python:3

ENV PYTHONDONTWRITEBYTECODE=1
ENV PYTHONUNBUFFERED=1

RUN apt-get update -y && apt-get upgrade -y

WORKDIR /opt/app

COPY . /opt/app/

RUN pip install --upgrade pip
RUN pip install -r requirements.txt

ENTRYPOINT ["bash", "docker-entrypoint.sh"]

```

Imagen 43. *Archivo Docker*
Fuente: *AWS*

Archivo Docker compose:

```yaml
services:
 web:
    build: .
      #entrypoint: build.sh
    command: python manage.py runserver 0.0.0.0:8000
    ports:
      - "8000:8000"
    environment:
      - POSTGRES_NAME=postgres
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=[redacted]
      - POSTGRES_HOST=database-postgresql-django-instance-1.c6nfjza8xbai.us-east-1.rds.amazonaws.com
    volumes:
      - web_app:/opt/app

volumes:
  web_app:
    name: django_app
```

Imagen 44. *Archivo Docker Compose*
Fuente: *AWS*

Además de esto vamos a ejecutar un script como entrypoint para realizar la configuración de las imágenes con las migraciones de base de datos y demás:

```bash
#!/bin/bash
# exit on error
set -o errexit

echo "Flush the manage.py command it any"

while ! python manage.py flush --no-input 2>&1; do
  echo "Flusing django manage command"
  sleep 3
done

echo "Migrate the Database at startup of project"

#python manage.py collectstatic --no-input

# Wait for few minute and run db migraiton
while ! python manage.py migrate  2>&1; do
   echo "Migration is in progress status"
   sleep 3
done

echo "Django docker is fully configured successfully."

exec "$@"
```

Imagen 45. *Script Entrypoint*
Fuente: *AWS*

9. Teniendo estos archivos en la base de nuestro proyecto y teniendo todo lo anterior mencionado (instancia de base de datos con AWS RDS, instancia de una maquina en AWS EC2, entorno configurado, código base del proyecto en la instancia de EC2)

Instancia de base de datos:

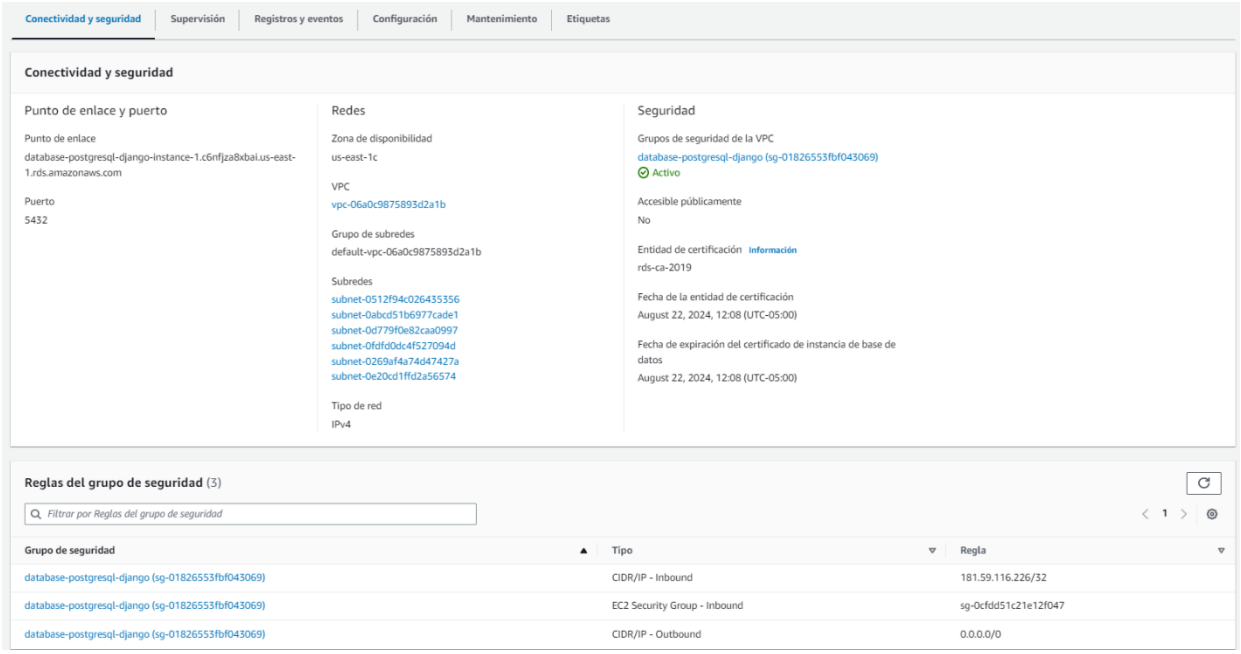


Imagen 46. *Instancia base de datos*
Fuente: *AWS*

Instancia de EC2:

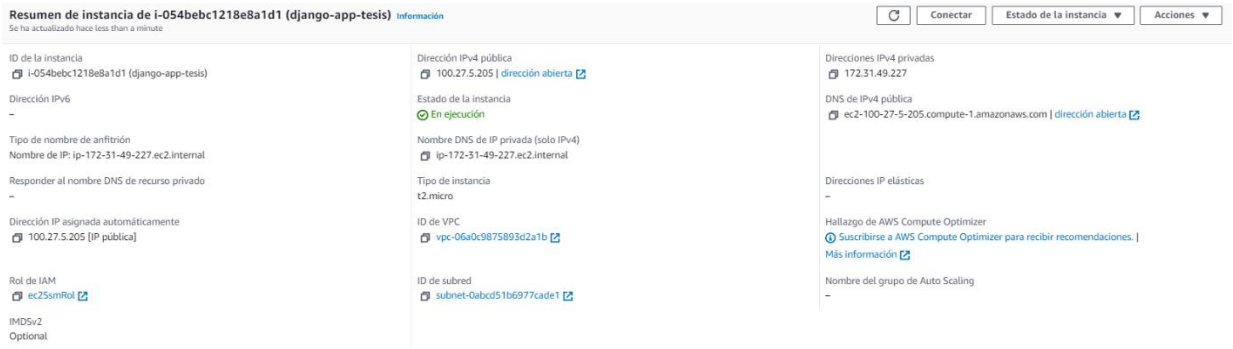


Imagen 47. *Instancia de EC2*
Fuente: *AWS*

Configuración del entorno:

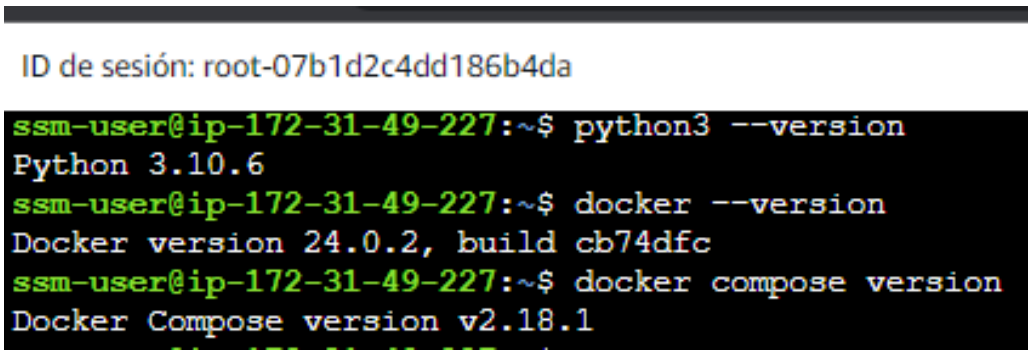


Imagen 48. *Configuración entorno*
Fuente: *AWS*

Código fuente del proyecto en la instancia de EC2:

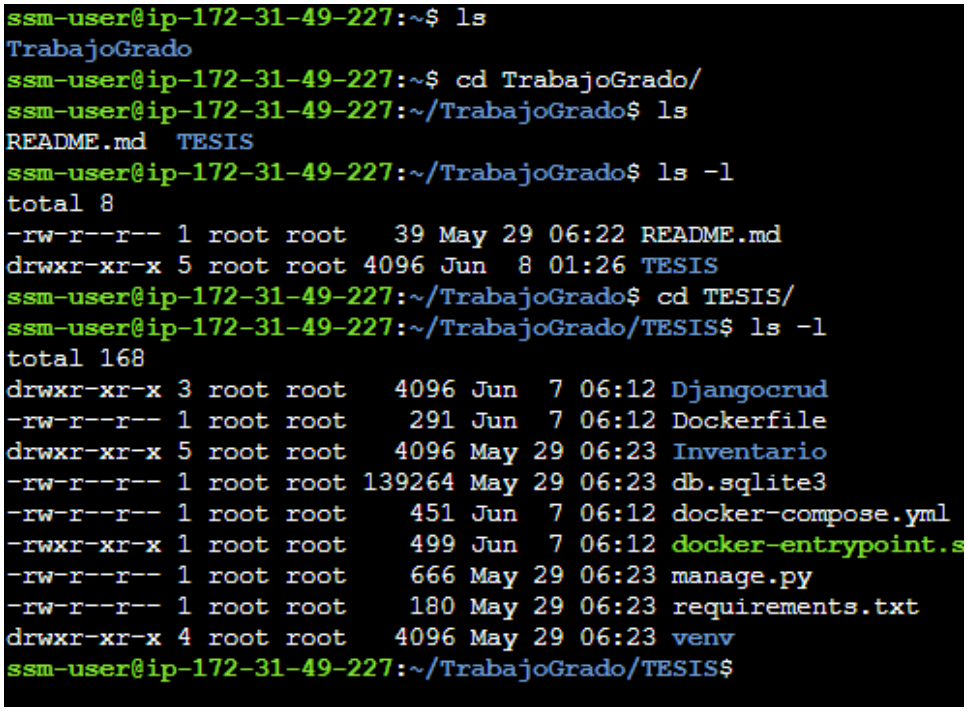


Imagen 49. *Código fuente*
Fuente: *AWS*

10. Teniendo ya todo esto vamos a realizar el despliegue de nuestro proyecto haciendo uso de Docker y Docker compose.

En el directorio base del proyecto en este caso “./TrabajoGrado/TESIS” podremos ejecutar el comando “sudo docker compose build” el cual utilizará el archivo de docker-compose.yml en la base del proyecto para construir la imagen de Docker con la aplicación y para la conexión a la instancia de AWS RDS para la base de datos:

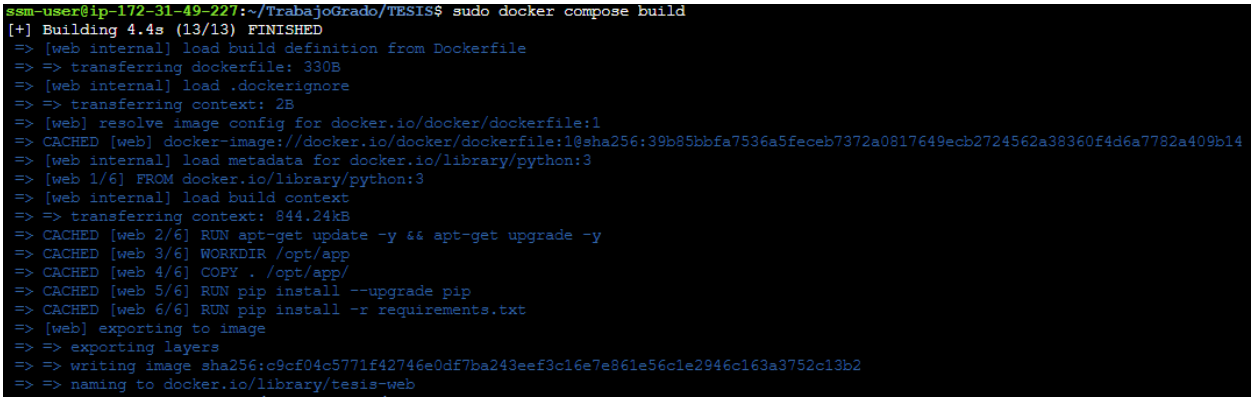


Imagen 50. *Docker compose en bash*
Fuente: *AWS*

11. Una vez construida la imagen podemos subir esta imagen para que el servidor de la aplicación se inicie, para lo cual es necesario que la instancia de base de datos esté corriendo y estén correctamente configuradas las credenciales para la conexión en el archivo de “docker-compose.yml”.

    Para levantar el proyecto vamos a ejecutar el comando “sudo docker compose up”

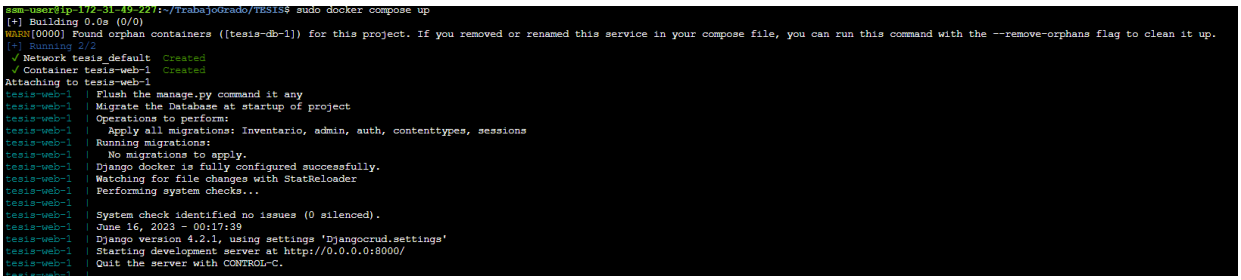


Imagen 51. *Comando Docker compose up*
Fuente: *AWS*

12. Una vez realizado este proceso podemos ver en los logs que la aplicación ha sido levantada la imagen en el puerto 8000, entonces lo que haremos es ir a buscar en la información de la

instancia el DNS público con el que podemos acceder a nuestra aplicación y para eso tendremos que agregar nuestra ip en el apartado de seguridad en las reglas de entrada esto como acción preventiva de exceso de uso lo cual nos podría llegar a generar costos.

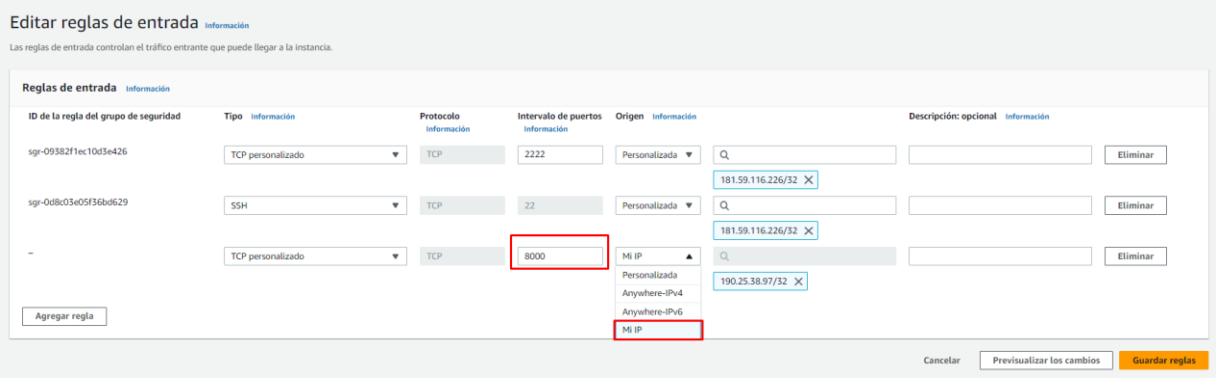


Imagen 52. *Edición reglas de entrada*
Fuente: *AWS*

Una vez agregada nuestra ip a las reglas de entrada podremos consumir nuestra aplicación de la siguiente manera:

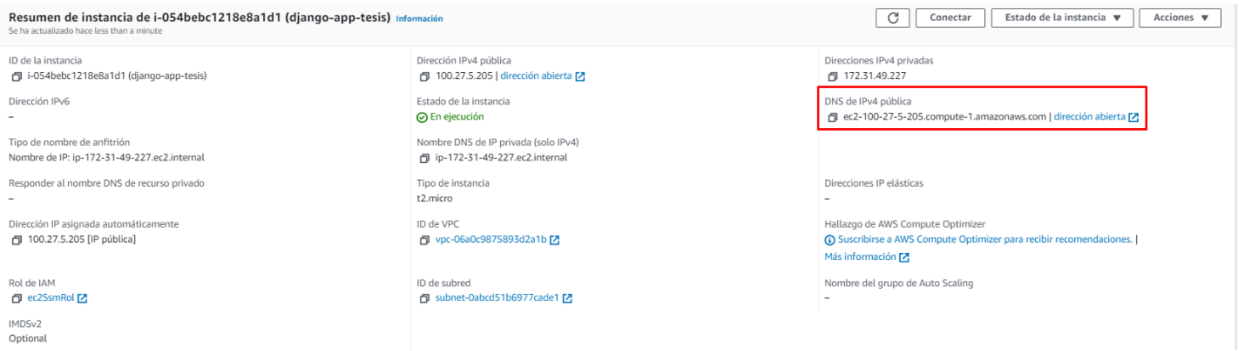


Imagen 53. *Resumen instancia*
Fuente: *AWS*

Tomando el DNS: http://ec2-100-27-5-205.compute-1.amazonaws.com/ y agregándole el puerto en el cual está nuestra imagen de Docker podremos acceder a nuestra aplicación y hacer uso de ella:

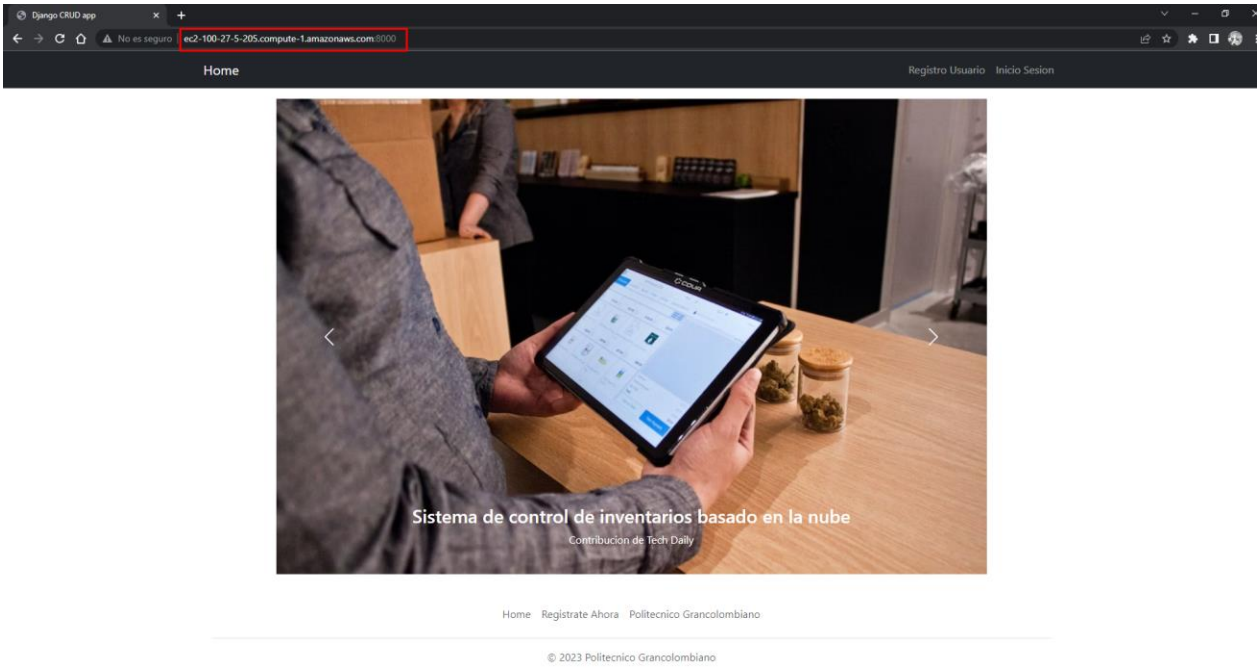


Imagen 54. *Página web*
Fuente: *AWS*

5.4.3 Despliegue en Render

Para realizar el despliegue en Render utilizando los servicios de Web service y PostgreSQL tenemos que configurarlo de la siguiente forma:

1. Web Service:
    - Se crea un servicio web en la opción

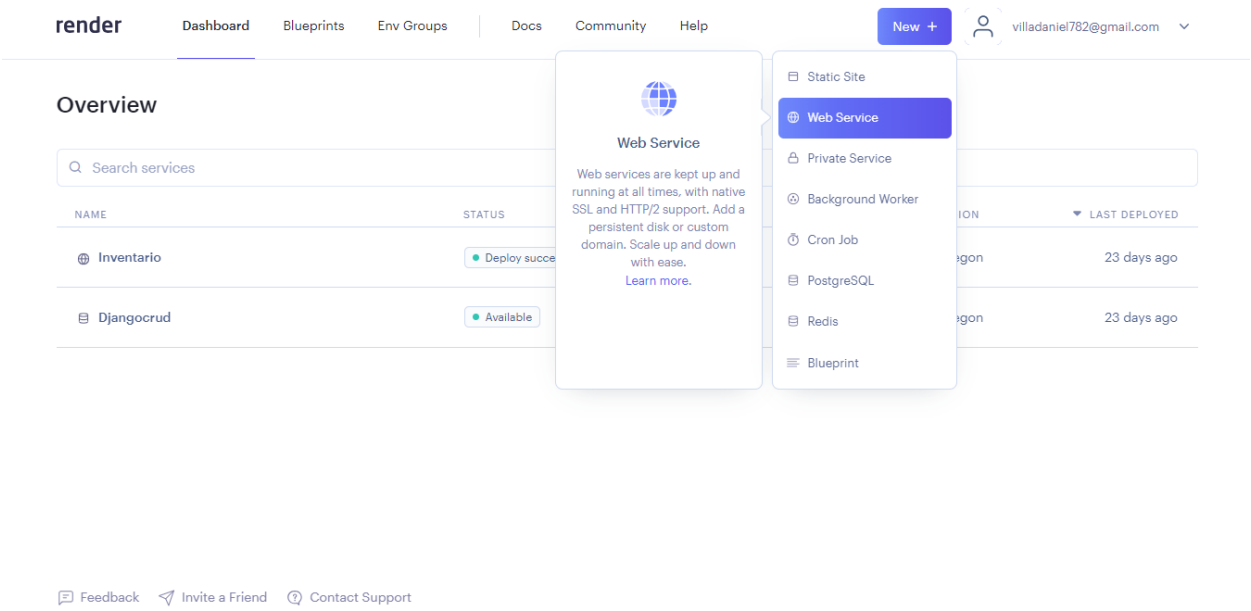


Imagen 55. *Creación de Web service*
Fuente: *Render*

- Luego con teniendo el repositorio en Github podremos realizar la conexión

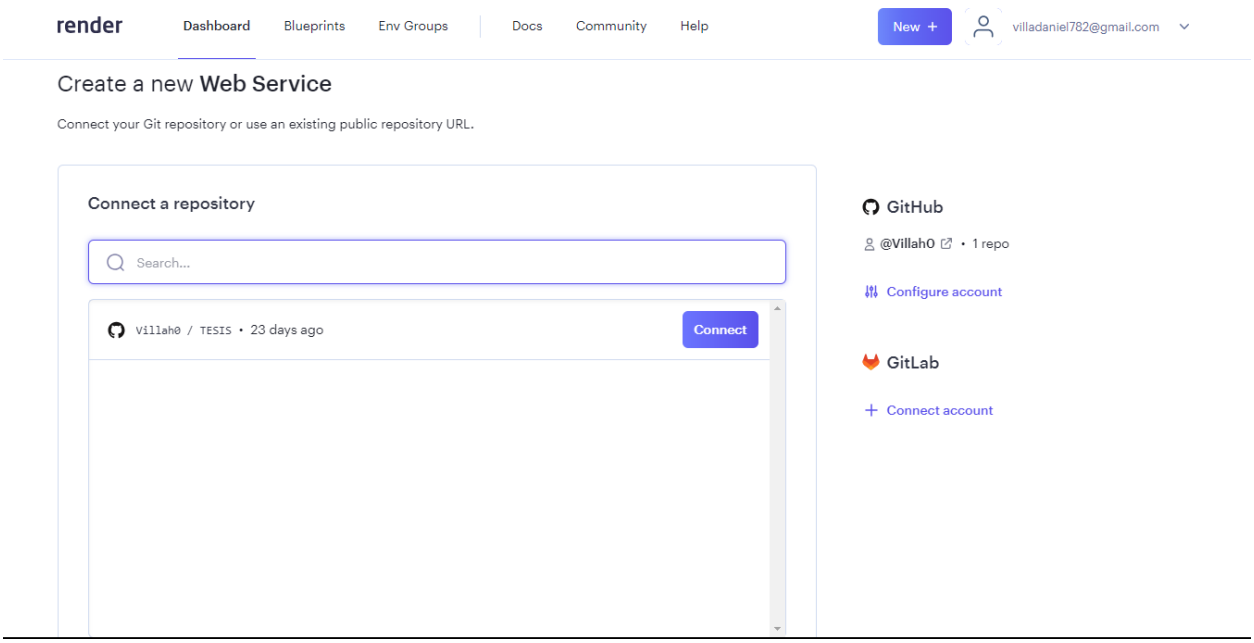


Imagen 56. *Creación de Web service*
Fuente: *Render*

- Siguiente paso será elegir el nombre del sitio Web, Región, la rama de conexión del repositorio, un comando que modificaremos más adelante y por último el plan si es gratuito o de pago.

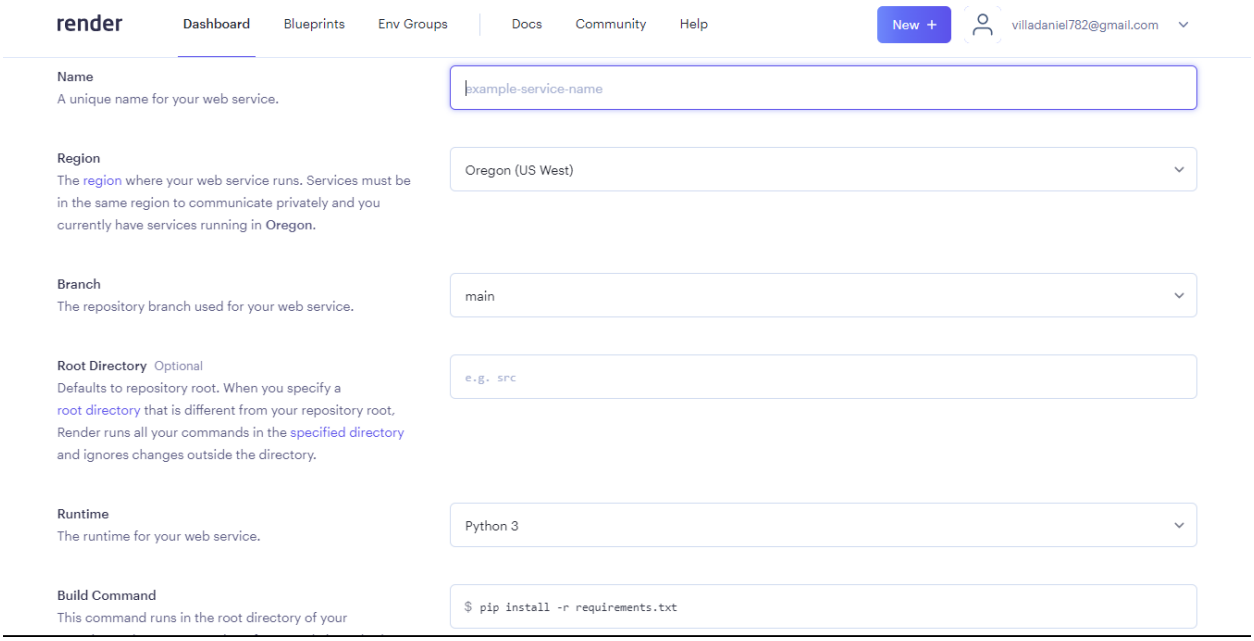


Imagen 57. *Creación de Web service*
Fuente: *Render*

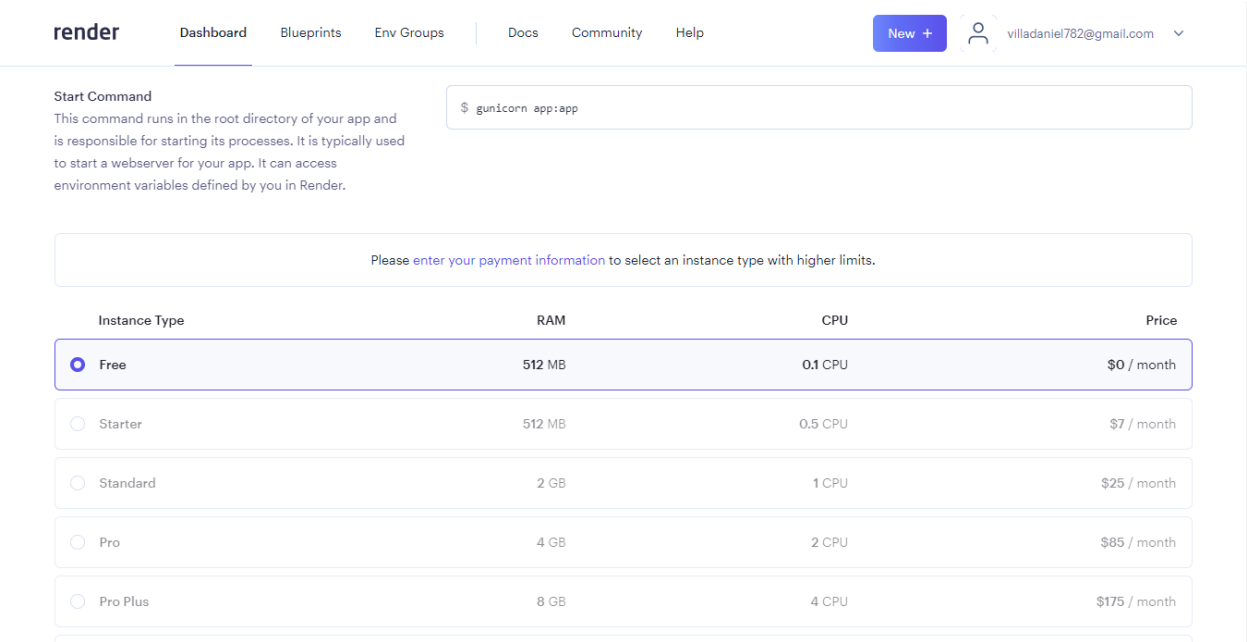


Imagen 58. *Planes de Web service*
Fuente: *Render*

- Los valores que pondremos serán

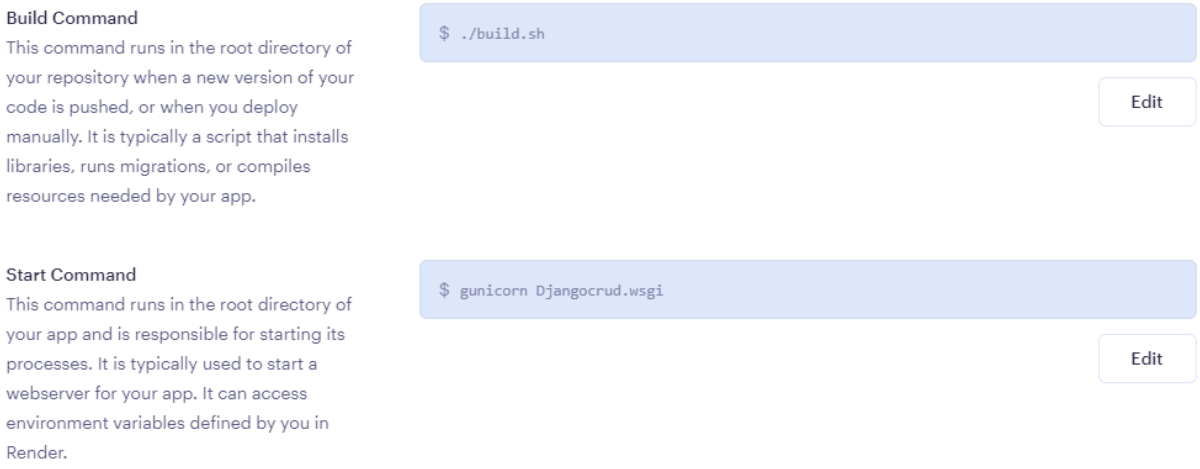


Imagen 59. *Valores de comando*
Fuente: *Render*

- Y ahora antes de realizar las modificaciones en las variables de entorno tenemos que cambiar el servicio de PostgreSQL

2. PostgreSQL:
    - Solo tendremos que modificar los siguientes datos ya que las tablas las crear Django en el backend

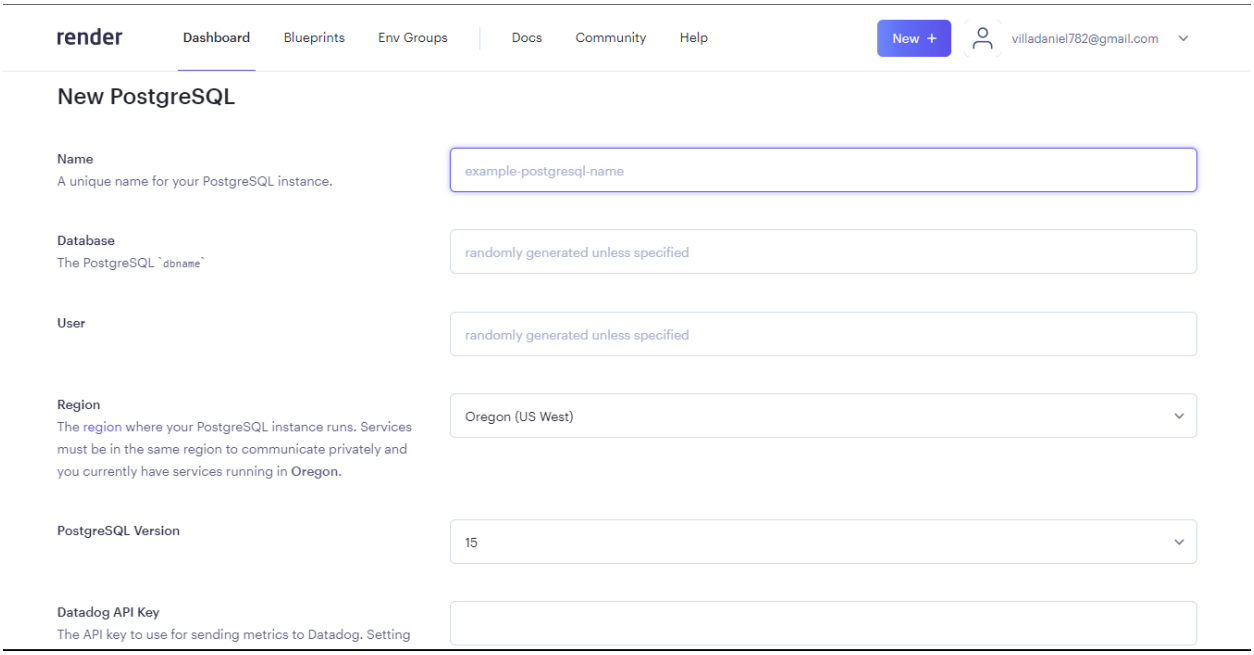


Imagen 60. *Creación base de datos*
Fuente: *Render*

- Al terminar nos quedaremos con una base de datos creada con estos datos

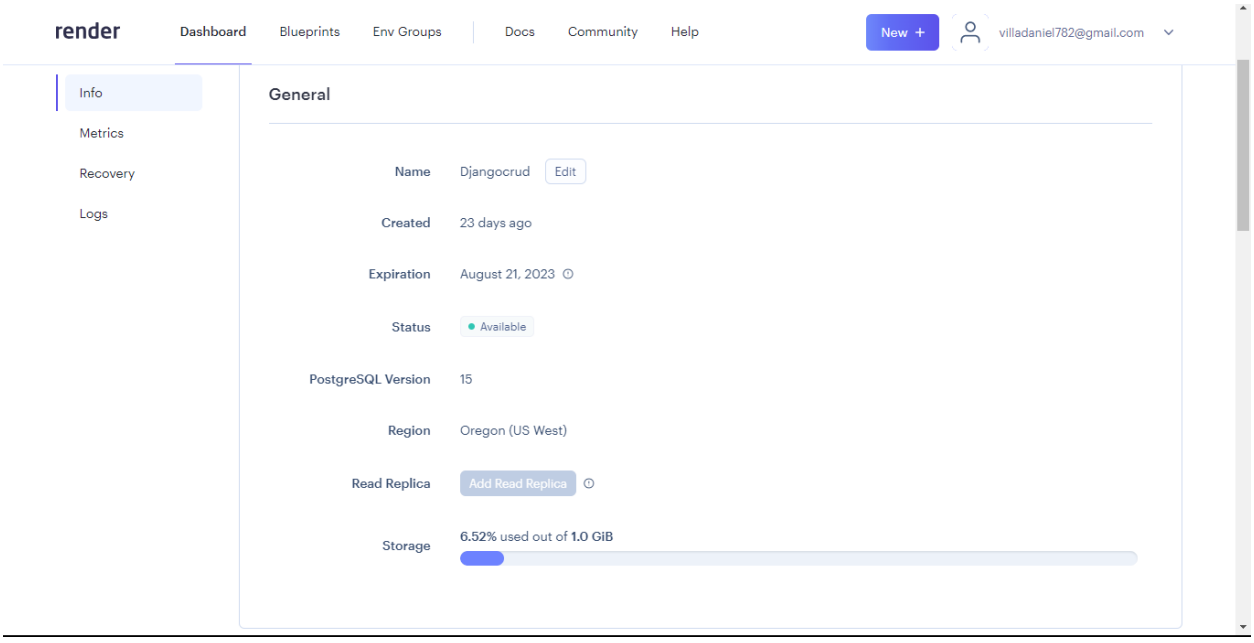


Imagen 61. *Información base de datos*
Fuente: *Render*

- Render nos proporcionara el servicio con ciertas conexiones

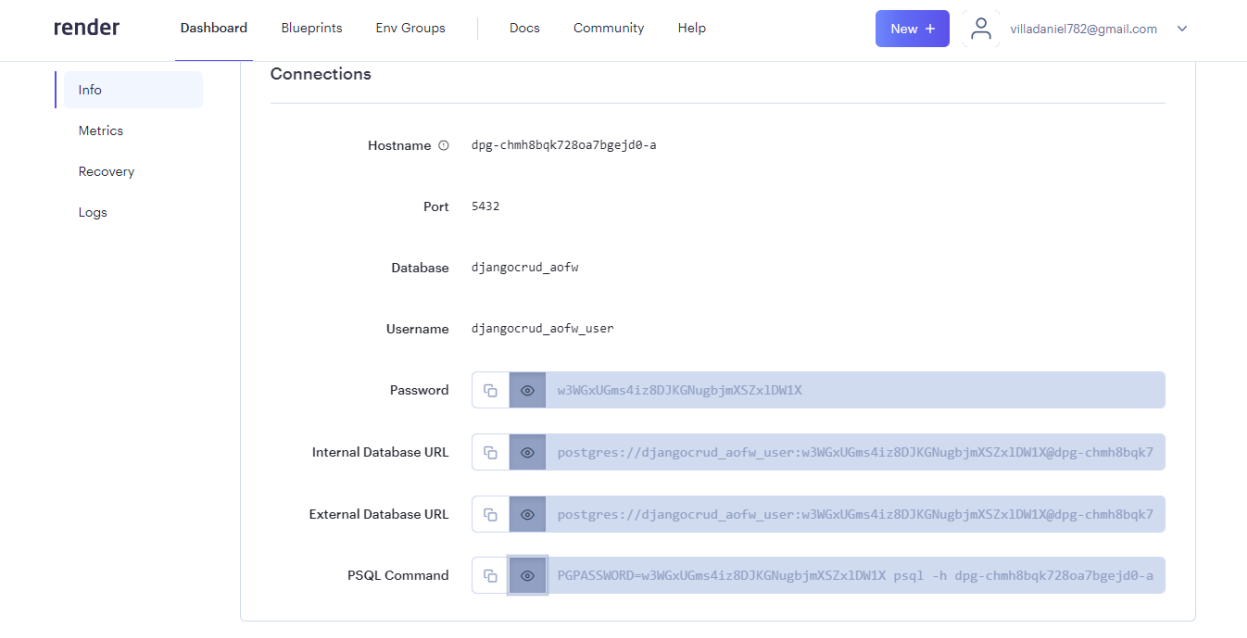


Imagen 62. *Información base de datos*
Fuente: *Render*

- Las cuales utilizaremos en el servicio Web para realizar su despliegue de esta forma:

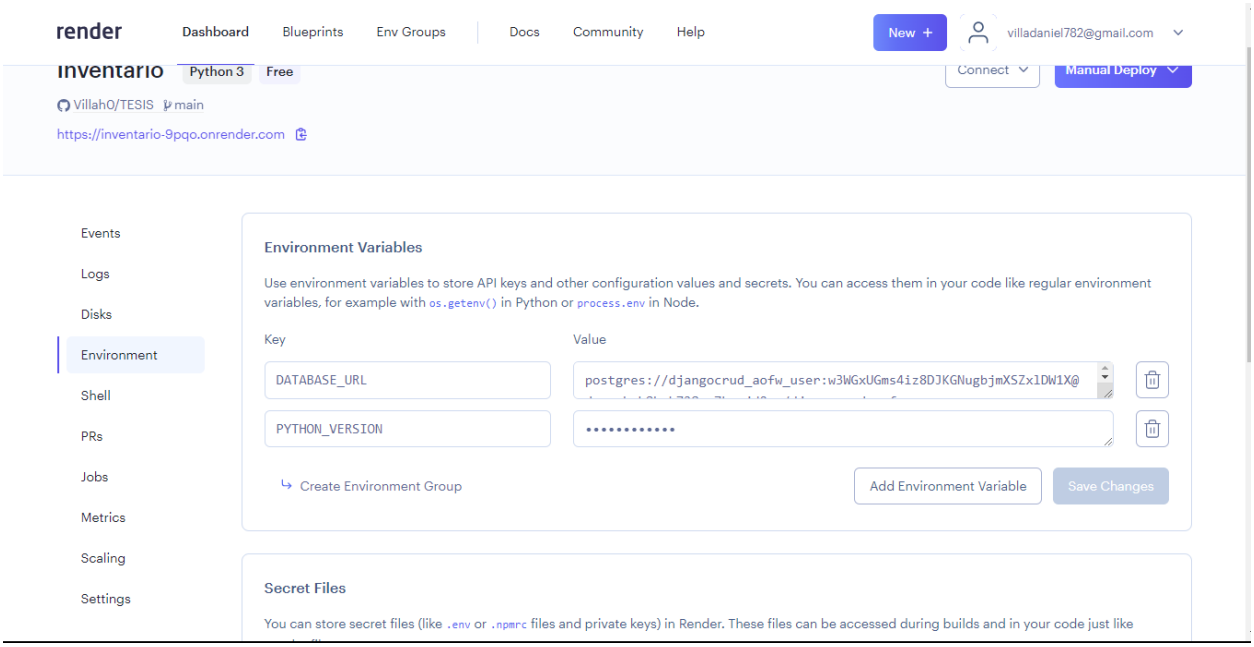


Imagen 63. *Valores de entorno*
Fuente: *Render*

- En las variables de entorno pondremos la versión de Python con la que se creó el backend y el enlace que nos proporciona el servicio de PostgreSQL para así realizar el despliegue y que Render nos brinde la url de la pagina

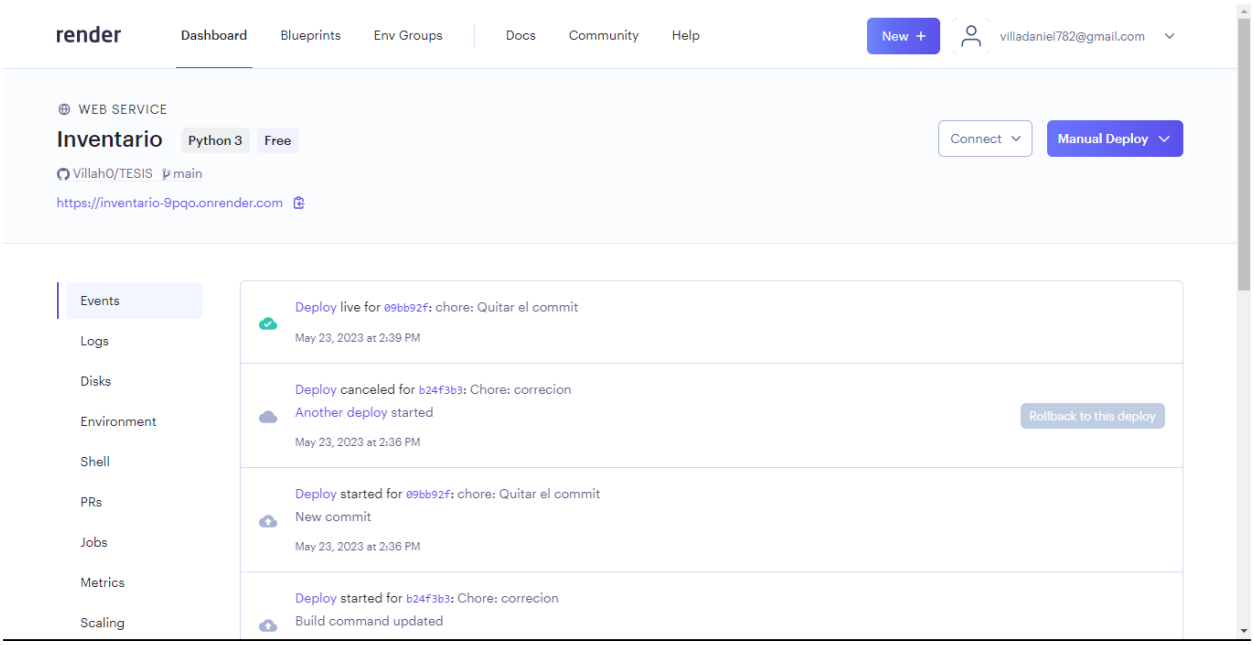


Imagen 64. *Despliegue Render*
Fuente: *Render*

Este despliegue se realiza cada vez que hay una modificación en la rama en la que se desarrolló el proyecto.

5.5.COMPARACIÓN ENTRE AWS Y RENDER

Característica	AWS – Capa gratuita	Render – Capa Gratuita
Conocimiento técnico necesario.	Se requiere buen conocimiento relacionado con: la administración de redes, administración de sistemas operativos y varias habilidades más que dependen de los requerimientos del sistema que se vaya a implementar.	Se requiere conocimientos técnicos de la nube no muy avanzados, ya que render nos ayuda en gran parte con las configuraciones complejas.
Servicios necesarios para la implementación.	Elastic Compute Cloud (EC2), Virtual Private Cloud (VPC), Relational Database Service (RDS), Identity and Access Management (IAM), System Manager	Web service, postgres SQL
Facilidad de implementación de CI/CD.	Debido a que en la capa gratuita no se cuenta con disponibilidad del servicio AWS Code commit se hacía más complicado montar un sistema de Continuous Integration / Continuous Deployment (CI/CD)	Dada la facilidad que nos ofrece render a la hora de integrar el repositorio del proyecto a sus servicios, render monta de manera autónoma y automática un proceso de CI/CD

Escalabilidad y rendimiento.	Debido a la gran variedad de servicios que nos ofrece AWS provee un rendimiento y una escalabilidad granular y personalizada, adicionalmente para las aplicaciones de alta demanda se ofrecen servicios dedicados al control y balanceo de la carga en múltiples zonas de disponibilidad.	Nos ofrece una manera sencilla de abarcar estos temas, haciendo de la escalabilidad una tarea automática, propia de render y un rendimiento óptimo para aplicaciones web y backend sin que se necesite de una configuración muy compleja.
Cobros generados.	Se tiene un enfoque de los costos basados en el uso, es decir las tarifas y los costos podrían variar dependiendo de los servicios utilizados y los recursos consumidos, cabe recalcar que en la implementación realizada no se han generado costos dado que se han utilizado únicamente los servicios cubiertos por la capa gratuita de AWS y no se han excedido los límites impuestos por esta.	Se tiene un enfoque más transparente de los costos teniendo tarifas fijas por recurso, simplificando y facilitando su entendimiento para muchos usuarios poco experimentados en el tema.
Flexibilidad y personalización.	Es más enfocado para requerimientos específicos dada que su flexibilidad es muy robusta, ofreciendo personalización para cada servicio y cada característica contratada, esto hace que se requiera una mayor experiencia en la configuración, pero mejora la personalización.	Es más enfocado a la simplificación y automatización lo que limita la personalización y ofreciendo poca flexibilidad, pero haciendo que no se requiera una gran experiencia para realizar las implementaciones.
Administración de la infraestructura.	Debido a la gran variedad de servicios que nos ofrece AWS para la configuración y administración de la infraestructura se tienen opciones avanzadas de red,	En gran parte es render quien se encarga de la administración de la infraestructura lo cual le quita la posibilidad al usuario de realizar una gestión y

	base de datos, almacenamiento, etc.	administración más personalizada, dejando todo en manos de los proveedores de nube.
Facilidad de uso.	Es una plataforma más compleja para su uso, pero más completa, lo que hace que se requiera de más tiempo y esfuerzo para entender y familiarizarse con sus servicios y configuraciones.	Simplificado y enfocado en que su uso sea intuitivo y fácil.

Tabla 3. *Cuadro comparativo entre Render y AWS*

Fuente: *Elaboración propia*

6. CONCLUSIONES Y TRABAJO A FUTURO

Se realizó el despliegue en Render.com y en AWS, se utilizó Render ya que tenía un manual de despliegue para cualquier ambiente de desarrollo Django, esto facilitaba la integración con las tecnologías utilizadas en el proyecto, por otra parte, traía un servicio de base de datos propia y todas estas funcionalidades eran de forma gratuita. Se utilizó AWS ya que es uno de los proveedores de nube más conocidos y robustos, dándonos una serie de servicios gratuitos y una capacidad de configuración y personalización para ajustar los servicios a las necesidades específicas, por lo tanto, al trabajar con la capa gratuita de Amazon Web Service y de Render debemos de tener en cuenta sus limitaciones.

Dado el ejercicio comparativo que se realizó entre el proveedor de nube “AWS” y el proveedor de nube “Render” se puede concluir lo siguiente:

- Debido a que la aplicación que se debía implementar no tenía una complejidad muy alta el proveedor que más se ajusta a estas necesidades fue render, ya que, al no necesitar requerimientos específicos de administración o personalización de los servicios de nube utilizados, en AWS sería más costoso realizar la implementación sin sacar mayor provecho de todas sus virtudes, por otro lado, en Render nos ofrecen mejores tarifas al pensar en una implementación real de la aplicación desarrollada.
- En el caso hipotético en el que se requiera una mayor personalización tanto en los servicios de red, almacenamiento y base de datos, una mayor disponibilidad y un mayor flujo de usuarios, en este caso sería mucho mejor el proveedor de nube AWS, debido a todas las virtudes que este nos ofrece para ese caso de uso en específico.

En cuanto al trabajo a futuro teniendo en cuenta que se podrían generar ciertos costos, estos serían:

- Manejar servidores multirregión y servicios en varias zonas de disponibilidad
- Uso de balanceadores de carga y balanceadores aplicación, haciendo uso de AWS ELB
- Hacer uso de la capa no gratuita de Render para así tener mejores prestaciones
- Añadir nuevas funcionalidades en el prototipo para que cubra mas necesidades de los usuarios

7. BIBLIOGRAFIA

1. Kaur, J., & Pahuja, R. (2021). Cloud-Based Inventory Management System: A Study. In Proceedings of the International Conference on Smart Technologies in Computer and Communication (pp. 421-429). Springer.

2. Khosravi, A., Ghandehariun, A., & Jannati, M. (2020). Inventory Management in Small and Medium-Sized Enterprises: A Case Study in Iran. In Handbook of Research on Small and Medium Enterprises in Developing Countries (pp. 329-345). IGI Global.

3. Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures (Doctoral dissertation, University of California, Irvine).

4. Khattak, A. M., Ali, A., & Khan, A. (2016). Comparative Analysis of RESTful Web Service Approaches in Cloud Computing. International Journal of Computer Applications, 138(1), 7-14.

5. Lee, S., & Kim, Y. (2019). A Study on the Development of Inventory Management System Based on Cloud Computing. Journal of the Korea Society of Computer and Information, 24(3), 57-65.

6. Chou, T., Chen, T., & Chang, H. (2012). Cloud computing: From beginning to end. Journal of Internet Technology, 13(4), 501-512. doi: 10.6138/JIT.2012.13.4.01

7. Aleti, A., & Singh, M. P. (2020). Cloud Computing Adoption in Small and Medium-sized Enterprises: An Exploratory Study. Journal of Global Information Technology Management, 23(2), 89-111. doi: 10.1080/1097198x.2018.1555285

8. K., & Li, C. (2018). Cloud computing adoption: A review and future research directions. International Journal of Information Management, 38(1), 78-101. doi: 10.1016/j.ijinfomgt.2017.09.002

9. Guan, L., Zhou, M., & Li, W. (2020). The relationship between cloud computing adoption and firm performance: Evidence from Chinese SMEs. Journal of Business Research, 116, 413-421. doi: 10.1016/j.jbusres.2020.07.042

10. Lee, J., Kozar, K. A., & Larsen, K. R. (2017). The technology acceptance model: Past, present, and future. Communications of the Association for Information Systems, 40(1), 1-13. doi: 10.17705/1CAIS.04001

11. Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., & Ghalsasi, A. (2011). Cloud computing – The business perspective. Decision Support Systems, 51(1), 176-189. doi: 10.1016/j.dss.2010.12.007

12. Chang, V., Ramachandran, M., & Li, C. (2018). Cloud computing adoption: A review and future research directions. International Journal of Information Management, 38(1), 78-101. doi: 10.1016/j.ijinfomgt.2017.09.002

13. García, A., Ramírez, M., & Castillo, E. (2017). Sistema de inventarios en la nube para la gestión de pequeñas y medianas empresas. Revista Tecnológica-Espejo, 10(1), 63-72.

14. Huamaní, R., Díaz, D., & Vargas, K. (2019). Implementación de un sistema de inventarios en la nube para la gestión de pequeñas empresas. Revista de Investigación en Tecnologías de la Información y Comunicación, 8(1), 1-9.

15. Shafiei, M., & Shafiei, M. (2020). Cloud-based inventory management system for small and medium enterprises. Journal of Cloud Computing: Advances, Systems and Applications, 9(1), 1-14.

16. Wang, Y., & Li, Y. (2018). Design and implementation of inventory management system based on cloud computing. Journal of Physics: Conference Series, 1029(1), 1-7.

17. Amazon Web Services. (2023a). ¿Qué es Docker? https://aws.amazon.com/es/docker/

18. Amazon Web Services. (2023b). Zonas de disponibilidad y regiones. https://aws.amazon.com/es/about-aws/global-infrastructure/regions_az/

19. Cloudflare. (2023). ¿Qué es una subred? https://www.cloudflare.com/es-es/learning/network-layer/what-is-a-subnet/

20. DiagramasUML. (s/f). Todos los diagramas UML. https://diagramasuml.com/

21. Django. (s/f). Django: The web framework for perfectionists with deadlines. https://www.djangoproject.com/

22. KeepCoding Team. (2023, abril 17). ¿Qué es Docker Compose? https://keepcoding.io/blog/que-es-docker-compose/#:~:text=Docker%20Compose%20es%20una%20herramienta,de%20forma%20f%C3%A1cil%20y%20r%C3%A1pida.

23. Omg. (2009). An OMG ® Unified Modeling Language ® Publication OMG ® Unified Modeling Language ® (OMG UML ®) OMG Document Number: Date. https://www.omg.org/spec/UML/20161101/PrimitiveTypes.xmi

24. Red Hat. (2018, marzo 19). ¿Qué es Linux? https://www.redhat.com/es/topics/linux

25. SafetyCulture. (2023, mayo 18). Control de inventarios. https://safetyculture.com/es/temas/manejo-de-inventario/control-de-inventarios/

26. Salas-Zárate, M. D. P., Alor-Hernández, G., Valencia-García, R., Rodríguez-Mazahua, L., Rodríguez-González, A., & López Cuadrado, J. L. (2015). Analyzing best practices on Web development frameworks: The lift approach. En Science of Computer Programming (Vol. 102). https://doi.org/10.1016/j.scico.2014.12.004

27. Vidal-Silva, C. L., Sánchez-Ortiz, A., Serrano, J., & Rubio, J. M. (2021). Experiencia académica en desarrollo rápido de sistemas de información web con Python y Django. Formacion Universitaria, 14(5). https://doi.org/10.4067/S0718-50062021000500085

28. Amazon Web Services. (n.d.). Administración de acceso con ACL - Amazon Simple Storage Service. Retrieved June 14, 2023, from https://docs.aws.amazon.com/es_es/AmazonS3/latest/userguide/acls.html

29. Amazon Web Services. (n.d.). AWS | Elastic compute cloud (EC2) de capacidad modificable en la nube. Retrieved June 14, 2023, from https://aws.amazon.com/es/ec2/

30. Amazon Web Services. (n.d.). Base de datos sin servidor - Amazon Aurora Serverless - AWS. Retrieved June 14, 2023, from https://aws.amazon.com/es/rds/aurora/serverless/

31. Amazon Web Services. (n.d.). Conexión a Internet mediante una puerta de enlace de Internet - Amazon Virtual Private Cloud. Retrieved June 14, 2023, from https://docs.aws.amazon.com/es_es/vpc/latest/userguide/VPC_Internet_Gateway.html

32. Amazon Web Services. (n.d.). Configurar tablas de enrutamiento - Amazon Virtual Private Cloud. Retrieved June 14, 2023, from https://docs.aws.amazon.com/es_es/vpc/latest/userguide/VPC_Route_Tables.html

33. Amazon Web Services. (n.d.). Grupos de seguridad - Amazon Virtual Private Cloud. Retrieved June 14, 2023, from https://docs.aws.amazon.com/es_es/vpc/latest/userguide/security-groups.html

34. Amazon Web Services. (n.d.). Internet-facing Classic Load Balancers - Elastic Load Balancing. Retrieved June 14, 2023, from https://docs.aws.amazon.com/es_es/elasticloadbalancing/latest/classic/elb-internet-facing-load-balancers.html

35. Amazon Web Services. (n.d.). Internet-facing Classic Load Balancers - Elastic Load Balancing. Retrieved June 14, 2023, from https://docs.aws.amazon.com/es_es/elasticloadbalancing/latest/classic/elb-internet-facing-load-balancers.html

36. Amazon Web Services. (n.d.). ¿Qué es Amazon VPC? - Amazon Virtual Private Cloud. Retrieved June 14, 2023, from https://docs.aws.amazon.com/es_es/vpc/latest/userguide/what-is-amazon-vpc.html

37. Amazon Web Services. (n.d.). ¿Qué es CIDR? - Explicación de los bloques y la notación CIDR - AWS. Retrieved June 14, 2023, from https://aws.amazon.com/es/what-is/cidr/

38. Mordor Intelligence. (n.d.). RETAIL CLOUD MARKET SIZE & SHARE ANALYSIS - GROWTH TRENDS & FORECASTS (2023 - 2028), from https://www.mordorintelligence.com/industry-reports/retail-cloud-market.