PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ESPECIALIDAD DE INGENIERÍA INFORMÁTICA

Sistema de Laboratorios PUCP

Documento de Arquitectura

Versión 2.0

PROPUESTO POR: Luis Flores luis.flores@pucp.edu.pe

ELABORADO POR:

Historial de Revisiones

Historial de revisiones				
Ítem	**Fecha**	**Versión**	**Descripción**	**Equipo**
1		1.0	Versión inicial.	●
2		2.0	Corrección de Diagramas de Componentes y Paquetes	●

Índice

1 Introducción

1.1 Propósito

El propósito de este documento es el dar una visión de la arquitectura del Sistema de Laboratorios Pucp. Se tiene como objetivo presentar la estructuración de alto nivel de nuestro sistema y sus propiedades globales.

1.2 Alcance

El documento de arquitectura nos proveerá la arquitectura del Sistema de Laboratorios Pucp. Dicho sistema facilitará la administración de los datos de las pruebas que realizan los laboratorios de la PUCP.

Este documento ha sido creado usando como base la información obtenida durante la etapa de análisis y diseño. Varios de los diagramas mostrados son el resultado de dichas etapas.

1.3 Definiciones, Acrónimos y Abreviaturas

1.3.1 ACID

Acrónimo de atomicidad, consistencia, insularidad y durabilidad por sus siglas en inglés, es un conjunto de características necesarias para las transacciones en las bases de datos.

1.3.1.1 Atomicidad

Indica a que las transacciones deben estar compuestas por operaciones indivisibles. Todas deben ejecutarse de manera correcta, de lo contrario se debe deshacer y regresar al estado anterior.

1.3.1.2 Consistencia

Consiste en que toda transacción en la base de datos debe empezar y terminar en un estado válido. Esto nos permite verificar que la base de datos mantenga la integridad, ofreciendo datos exactos y válidos.

1.3.1.3 Insularidad

Esta propiedad nos asegura que ninguna operación en la base de datos intervendrá con otra, asegurando que las transacciones en paralelo se realizarán con éxito.

1.3.1.4 Durabilidad

Hace referencia a la persistencia de las transacciones, es decir, que una vez que sean realizadas no podrán deshacerse aunque el sistema presente un fallo.

1.3.2 HTML

'Hypertext markup language', lenguaje para la elaboración de páginas web bajo una estructura definida que muestra el texto e imágenes, entre otros que la aplicación posea.

1.3.3 HTTP

'Hypertext transfer protocol' o protocolo de transferencia de hipertexto, permite la transferencia de información a través del internet .

1.3.4 REST

Abreviatura de 'representational state transfer' es una arquitectura ejecutada sobre HTTP para la transferencia de datos mediante el internet. Esta cuenta con 2 métodos importantes:

- GET: Utilizado para obtener información del servidor sin realizar modificaciones.
- POST: Método que permite crear un recurso para la modificación del servidor .

1.3.5 Django

Framework de código abierto escrito en Python que implementa el diseño MTV para el desarrollo de aplicaciones web.

1.3.6 MTV

Abreviatura de 'Model-template-view', arquitectura empleada por Django.

- Model: Capa contenedora del modelo de datos a utilizar en la aplicación.
- View: Capa encargada de la administración de las vistas que serán utilizadas por el usuario.
- Template: Capa que representa directamente a los datos.

1.3.7 API

'Application programming interface', es la representación de un conjunto de llamadas de bibliotecas que han sido desarrolladas por un software externo para su uso, esta es una capa de abstracción.

1.3.8 jQuery

Biblioteca que permite la interacción entre páginas HTML y la interacción mediante AJAX.

1.3.9 AJAX

Tecnología utilizada para la comunicación entre el cliente de manera asíncrona con el servidor.

1.4 Visión General del documento

El actual documento consta principalmente de la representación de la arquitectura del Sistema de Laboratorios Pucp. Se ha escogido el modelo 4+1, por lo cual primero se presentará una sección donde se explique sobre dicho modelo y sus elementos. En las siguientes secciones se verán los elementos del modelo de nuestra arquitectura. Además del modelo 4+1, se cuenta con información sobre metas, restricciones, tamaño, rendimiento y calidad del sistema propuesto.

2 Representación de la arquitectura

Para representar nuestra arquitectura usaremos el modelo 4+1 de Philippe Kruchten. Dicho modelo está conformado por 5 vistas, las cuales se pueden apreciar en el siguiente gráfico:

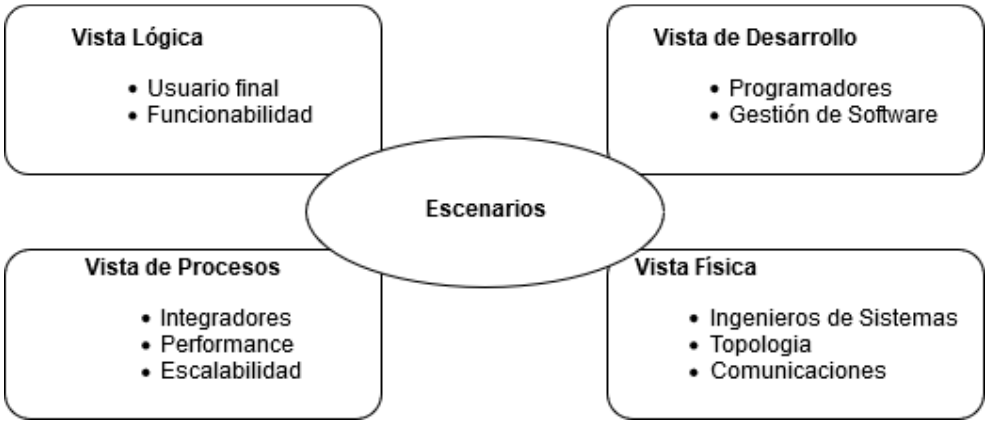

Figura 1. Modelo 4+1

A continuación explicamos en qué consiste cada vista:

- **Vista lógica**
  Se centra en apoyar los requisitos funcionales, descomponiendo el sistema en un conjunto de llaves abstractas. Esta descomposición no solo ayuda en el análisis funcional, sino también sirve para identificar las mecánicas básicas y el diseño de los elementos del sistema.

- **Vista de procesos**
  Tomando en cuentas algunos de los requisitos no funcionales, la vista de procesos centra su atención en los problemas de concurrencia y distribución, de la integridad del sistema, la tolerancia a los errores y como las principales abstracciones de la vista lógica encajan dentro de la arquitectura de procesos.

- **Vista de desarrollo**
  Tiene un enfoque en la organización modular del software en su entorno de desarrollo. Se separa el software en diferentes partes para poder desarrollar cada parte por un grupo pequeño de programadores. Los subsistemas se organizan en una jerarquía de capas, donde cada capa provee una interfaz reducida y bien definida a las capas superiores.

- **Vista física**
  Toma en cuenta los requisitos no funcionales, tales como la disponibilidad, la tolerancia de errores, el performance y la escalabilidad. Se identifican varios elementos del software y son mapeados en nodos de hardware. Con dicho mapeo se puede apreciar la distribución de estos.

- **Escenarios**
  Se observa como los elementos de las 4 vistas trabajan juntos en un pequeño conjunto de escenarios importantes.

3 Metas y Restricciones de la arquitectura

Metas

- El software podrá realizar la trazabilidad de los servicios realizados.
- Se podrá acceder al software desde navegadores como Google Chrome, Firefox, Safari y Microsoft Edge.

Restricciones

- El sistema contará con un log de errores, el cual se almacenará en la base de datos.
- El sistema deberá ser configurable en base a parámetros predefinidos.

4 Vista de Casos de Uso

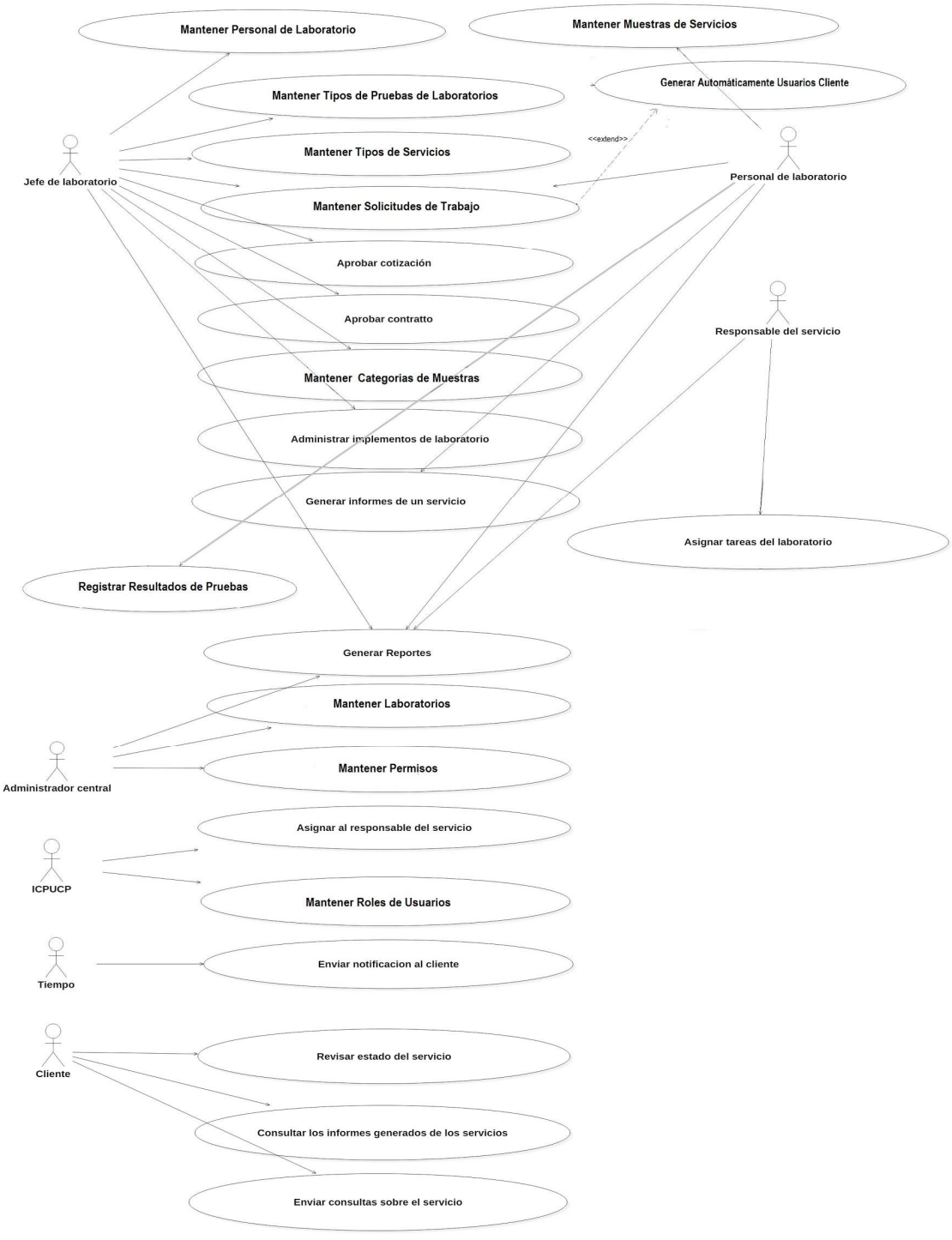

Figura 2. Casos de uso

5 Vista Lógica

La arquitectura seleccionada para el sistema está organizada en distintas agrupaciones lógicas, las cuales cuentan con un determinado propósito que se detalla a continuación:

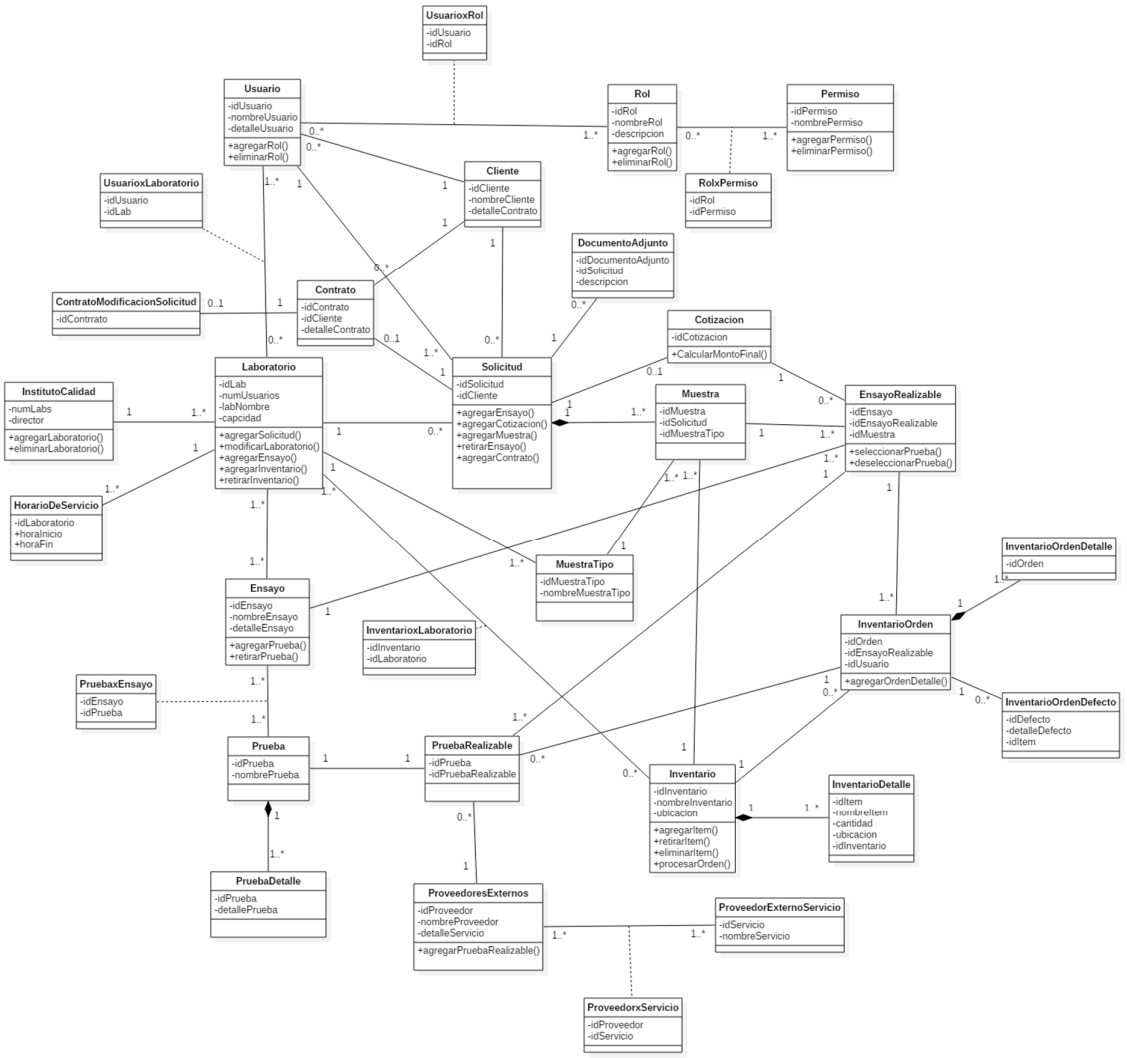

Figura 3. Diagrama de clases lógico

- **View**

  En esta sección se ejecuta el procesamiento “real” de la solicitud HTTP, es decir, es donde se realizan las verificaciones y procesamiento necesario de la solicitud para aplicar la lógica de negocios y responder a la solicitud. Para responder a la misma, se transfiere esta información a la sección *Template*, para mostrarla.

- **Template**

  Es la sección que determina la presentación de los datos generados por el *View*, en una aplicación web esta presentación consiste de contenido HTML, y será enviada al navegador para ser mostrada al usuario.

- **Model**

  Un *Model* es representativo de un objeto lógico de negocio, sirve como interfaz entre la aplicación y la base de datos, e implementa el patrón *Active Record* para permitir la persistencia en el sistema de base de datos.

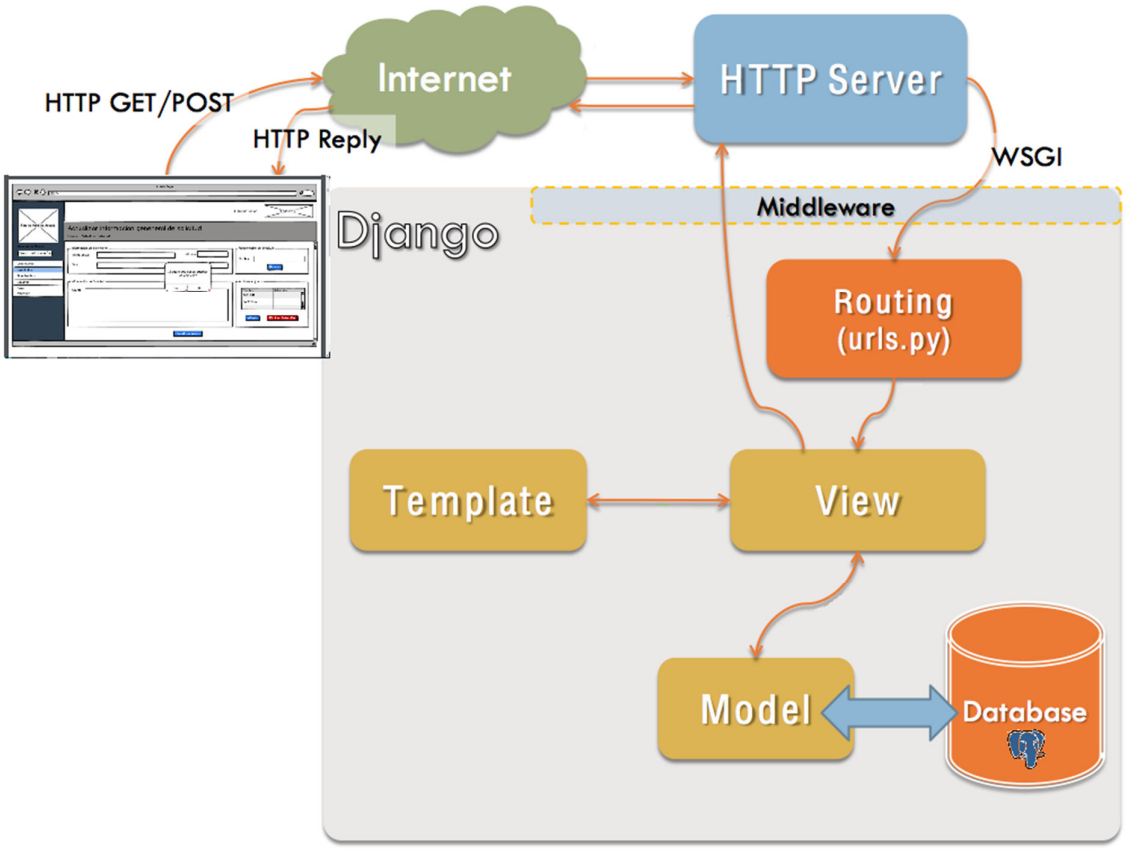

Figura 4. Esquema MTV (Model Template View)

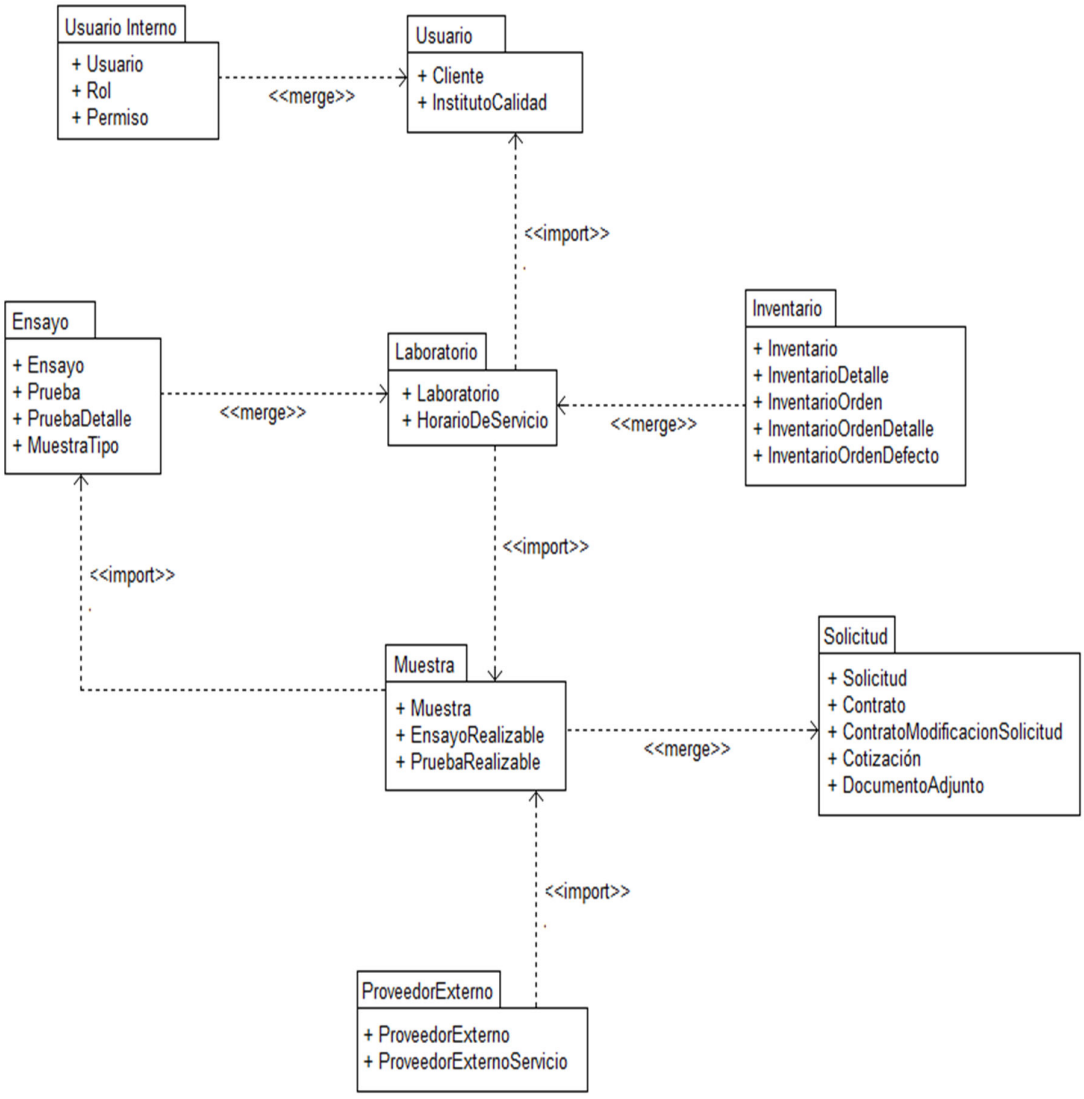

Figura 5. Diagrama de paquetes

6 Vista de Proceso

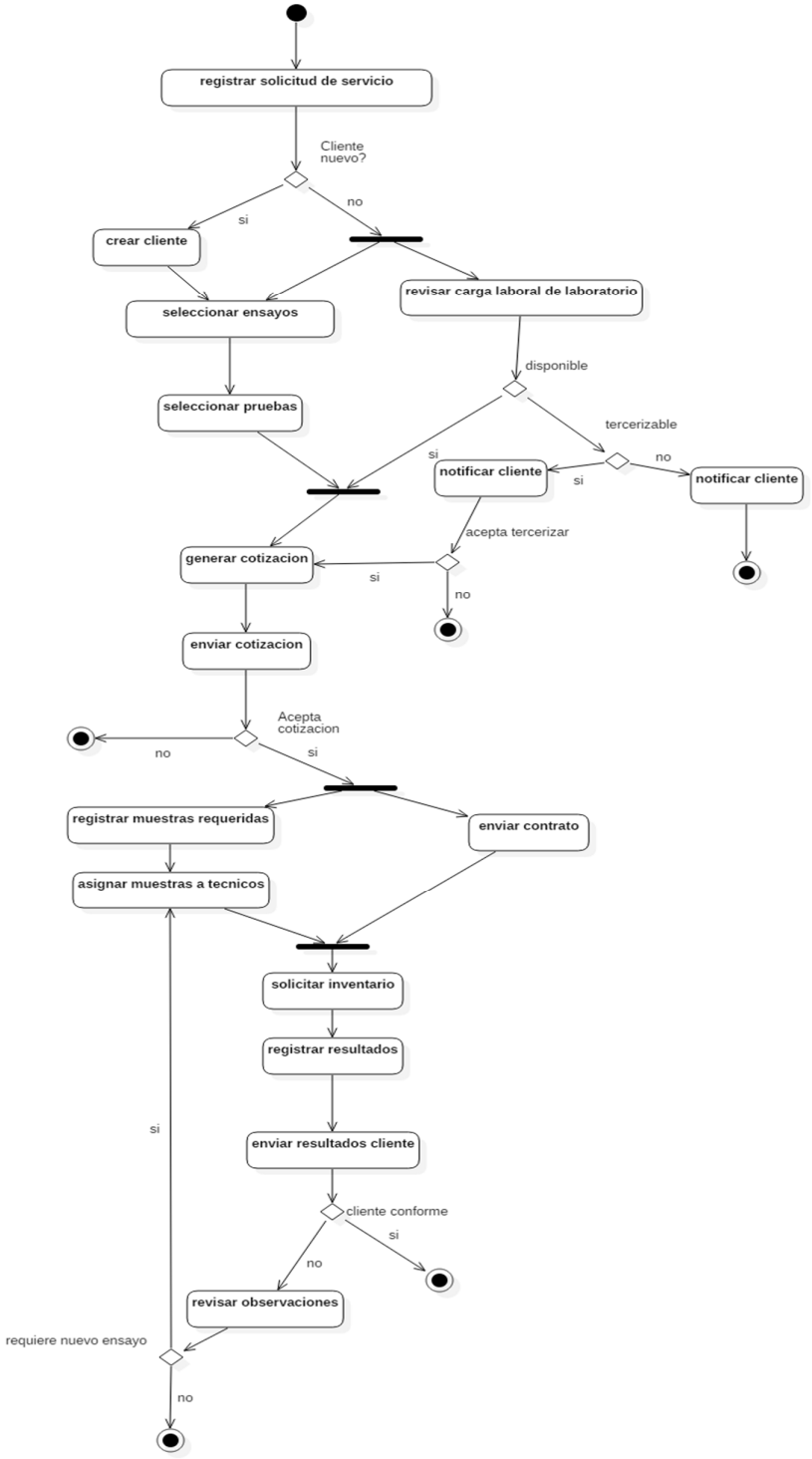

Figura 6. Diagrama de actividades

7 Vista de Despliegue

Los dispositivos que estarán conectados y en los que estará distribuido el software son los siguientes:

1. Cliente Web (Externo): Ordenador que accede a la página web del sistema mediante un navegador instalado en el dispositivo. La funcionalidad a la que se le asocia es a la del cliente para consultar el estado de sus pedidos.

2. Cliente Web (Interno): Ordenador que accede a la página web del sistema mediante un navegador instalado en el dispositivo. La funcionalidad a la que se le asocia es a la de los usuarios del laboratorio y otro personal interno a la PUCP para acceder y administrar la información respecto a los pedidos de los clientes.

3. Servidor Web (Apache): El servidor que contiene la página web y al que se conectan los clientes web para acceder.

4. Servidor de Aplicación (Django): El servidor que ejecuta el componente del software del sistema, es decir la lógica del negocio y el acceso a la base de datos.

5. Base de Datos (PostgreSQL): El servidor donde se almacena la base de datos administrada por el sistema.

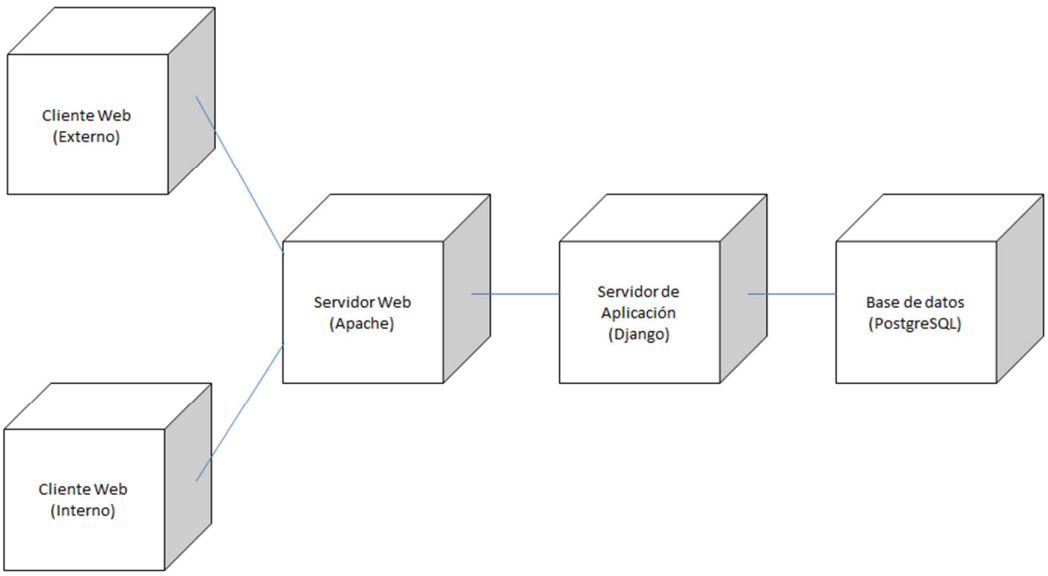

Figura 7. Diagrama de Despliegue

8 Vista de Implementación

8.1 Descripción

En esta parte se presenta una visión general de los paquetes a utilizar en el sistema, desde la presentación al cliente, pasando por las aplicaciones web y finalizando en la base de datos.

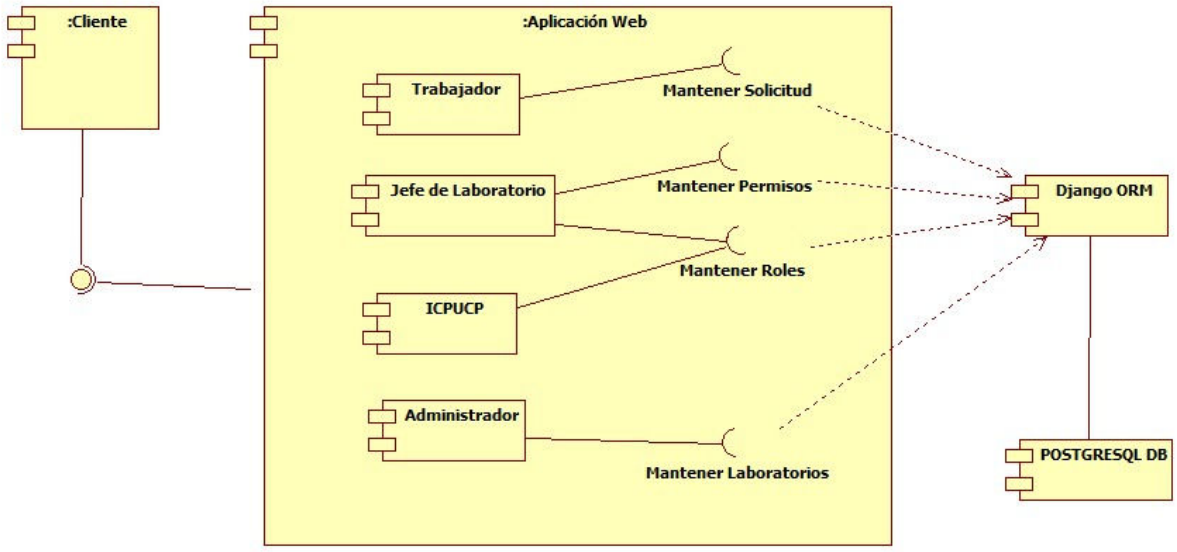

Figura 8. Diagrama de componentes

9 Tamaño y rendimiento

El sistema facilitará la administración de los datos de las pruebas que realizan los laboratorios de la PUCP. Por dicho motivo, en cuanto a tamaño y rendimiento, se puede decir lo siguiente:

- Se espera que haya 50 usuarios activos al mismo tiempo.
- Se creará un sistema lo más escalable posible.
- Los clientes no son usuarios que modifiquen nuestra base de datos, sino usuarios que se limitan a consultar. Debido a la necesidad de estar informado con datos precisos es necesario que la data almacenada cumpla dicha expectativa.

10 Calidad

Ciertos requerimientos de calidad que deberán cumplirse para la satisfacción del usuario son:

- Ser compatible con los navegadores Firefox, Chrome, Safari y Microsoft Edge.
- Enviar notificaciones periódicamente a los usuarios.
- Ser de fácil aprendizaje y poderse dominar rápidamente.
- Ser responsive.
- Tener un diseño minimalista para evitar la sobrecarga de información en la interfaz.

11 Referencias

FLORES, Luis

2017 "06 - Arquitectura". Consulta: 4 de Mayo de 2017

https://paideia.pucp.edu.pe/cursos/pluginfile.php/466289/mod_resource/content/3/Arquitectura.pdf

KRUCHTEN, Philippe

1995 "Architectural Blueprints—The "4+1" View Model of Software Architecture". IEEE Software 12 (6). pp 42-50. Consulta: 4 de Mayo de 2017

http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf