<MakeYourParty>
Software Architecture Document

Version <1.0>

Revision History

Date	Version	Description	Author
<08/Marzo/16>	<1.0>	<Architectural Goals and Constraints, Use-Case View, Logic View, and Size and Performance>	<Miguel Lopez, Miguel Luna, Fernando Díaz >

1. Introduction

[The introduction of the **Software Architecture Document** provides an overview of the entire **Software Architecture Document**. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of the **Software Architecture Document**.]

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

[This section defines the role or purpose of the **Software Architecture Document**, in the overall project documentation, and briefly describes the structure of the document. The specific audiences for the document is identified, with an indication of how they are expected to use the document.]

1.2 Scope

[A brief description of what the Software Architecture Document applies to; what is affected or influenced by this document.]

1.3 Definitions, Acronyms, and Abbreviations

[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Software Architecture Document**. This information may be provided by reference to the project's Glossary.]

1.4 References

[This subsection provides a complete list of all documents referenced elsewhere in the **Software Architecture Document**. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]

1.5 Overview

[This subsection describes what the rest of the **Software Architecture Document** contains and explains how the **Software Architecture Document** is organized.]

2. Architectural Representation

Logical View

Describe el diseño del modelo de objetos. También describe la realización de los casos de uso más significativos del sistema. **Diagrama de componentes UML**

Process View

Se enfoca más en los requerimientos no funcionales. Describe los aspectos de concurrencia y sincronización del diseño.

Implementation View

Se enfoca en los componentes de software. Describe las capas y subsistemas de la aplicación. **Diagrama de clases UML**

Deployment View

Se enfoca en la topología del sistema. Describe el mapeo del software en el hardware y muestra los aspectos distribuidos del sistema. **Diagrama de despliegue UML**

Use Case View

Describe un set de escenarios y/o casos de uso que representan funcionalidad significativa del sistema. **Diagrama de casos de uso UML**

3. Architectural Goals and Constraints

Esta aplicación tiene que ser portable ya que será desarrollada para Android y IOS y contará con un sitio web para la administración de contenido. Dentro de la app se realizarán pagos con tarjetas de crédito, débito o PayPal por lo que el requerimiento de seguridad es muy importante ya que contendrá datos sensibles. La clave fundamental del proyecto es el resguardo de la privacidad de los datos de los usuarios, ya que la intromisión de terceros en la información de los clientes tendría un impacto catastrófico en el negocio.

La experiencia de usuario también es considerada como un punto fundamental en el diseño de la aplicación. Los clientes deben de ser capaces de localizar cada una de las funcionalidades de la aplicación con el mínimo esfuerzo. Siendo así, la aplicación tiene como requisito estar apegada a las tendencias actuales de usabilidad de las aplicaciones móviles.

Las herramientas de desarrollo que se ocuparán para el desarrollo del proyecto son: Android Studio en el desarrollo de la app de android, XCode para el desarrollo de la app de IOS y CakePHP para el desarrollo del sitio administrador de contenido.

El equipo está constituido de 3 personas, una para cada etapa del proyecto (Android, IOS, web). El desarrollo de la aplicación está planeada para entregarse al cliente en un periodo de 6 a 8 meses. Empezando por el diseño de la base de datos, seguido del desarrollo del sitio administrador, servicios web para consumo de app, desarrollo de Android y desarrollo de IOS. Es decir se tendrán dos meses para el desarrollo de cada etapa y dos meses de pruebas.

4. Use-Case View

Usuario:

1. Login
2. Registro <- Carga Arquitectónica
3. Logout
4. Selecciona Fecha y tiempo de renta
5. Selecciona Terreno
6. Selecciona DJ
7. Selecciona Luces
8. Paga Fiesta <- Carga arquitectónica

Administrador:

1. Alta, baja cambios proveedores

Proveedor

1. Alta, baja y cambios productos

Los siguientes diagramas describen los casos de uso del sistema

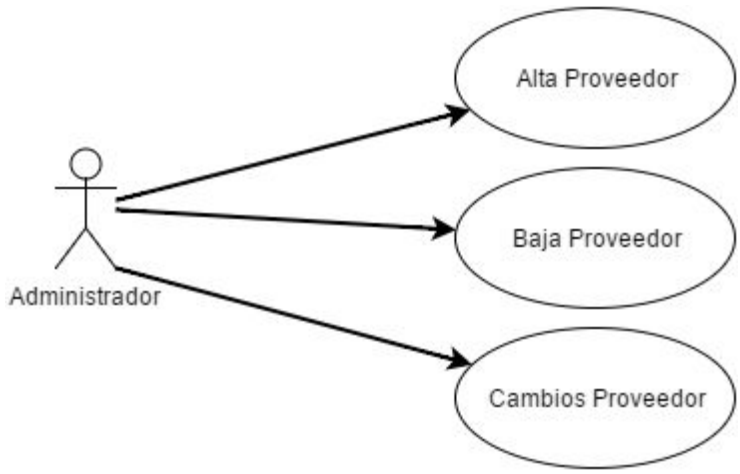


Figura 1 Casos de uso administrador

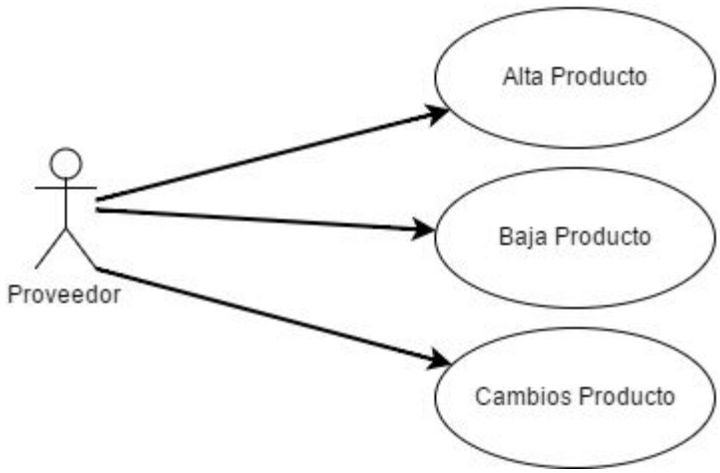


Figura 2 Casos de uso proveedor

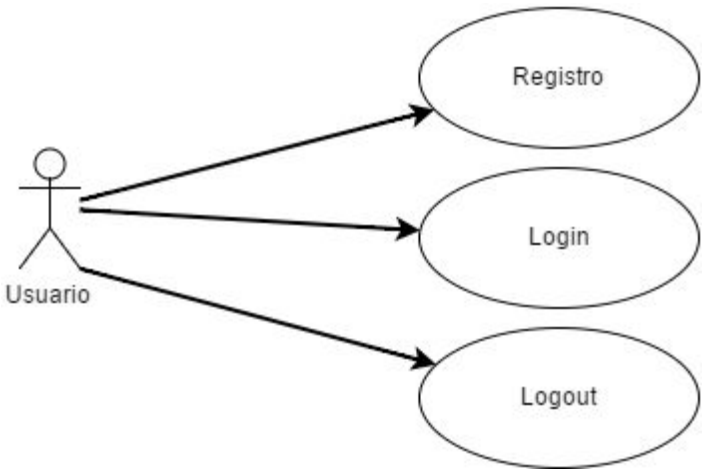


Figura 3 Casos de uso usuario parte 1

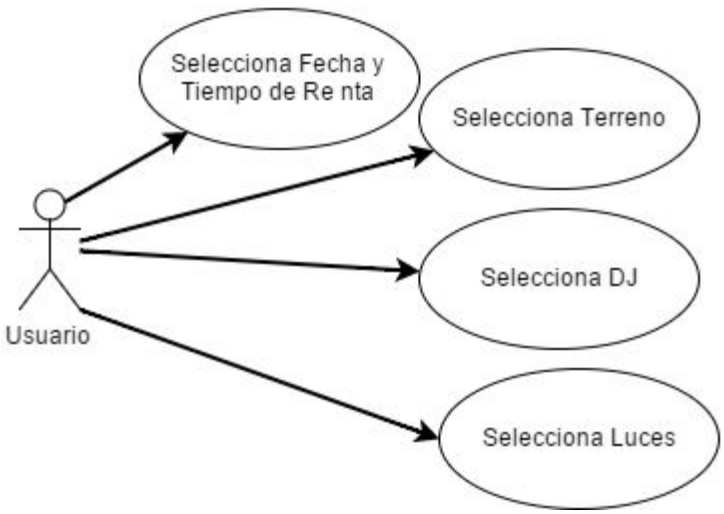


Figura 4 Casos de uso usuario parte 2

Figura 5 Casos de uso usuario parte 3

Descripción casos de uso

1. **Login**

El caso de uso toma lugar cuando un usuario quiere entrar a la aplicación, tendra que proporcionar su e-mail y contraseña, en caso de ser correctos entrara a la aplicación, en caso de ser incorrectos el sistema mostrará un mensaje de que no es posible entrar al sistema.

2. **Registro**

El caso de uso toma lugar cuando un usuario nuevo quiere registrarse a la aplicación, el usuario se registrará por medio de su email y una contraseña.

3. **Logout**

El caso de uso toma lugar cuando un usuario dentro de la aplicación quiere salir, podrá seleccionar el icono de Logout para salir del sistema.

4. **Selecciona Fecha y tiempo de renta**

El caso de uso toma lugar cuando un usuario está creando su fiesta, selecciona fecha y tiempo de renta del evento

5. **Selecciona Terreno**

El caso de uso toma lugar cuando un usuario seleccione la fecha y tiempo de renta se mostrarán los terrenos disponibles para esa fecha y tiempo de renta.

6. **Selecciona DJ**

   El caso de uso toma lugar cuando un usuario seleccione la fecha y tiempo de renta se
   mostrarán los DJs disponibles para esa fecha y tiempo de renta.

7. **Selecciona Luces**

   El caso de uso toma lugar cuando un usuario seleccione la fecha y tiempo de renta se
   mostrarán las luces disponibles para esa fecha y tiempo de renta.

8. **Paga Fiesta**

   El caso de uso toma lugar cuando un usuario seleccione una opción terreno, luces o DJ se desglosa el precio total y se procede a la carga de los servicios.

9. **Alta, baja cambios proveedores**

   El caso de uso toma lugar cuando un administrador está logueado en el sistema web el administrador podrá dar de alta, baja o modificar a los proveedores.

10. **Alta, baja cambios productos**

    El caso de uso toma lugar cuando un proveedor está logueado en el sistema web el proveedor podrá dar de alta, baja o modificar sus servicios a ofrecer.

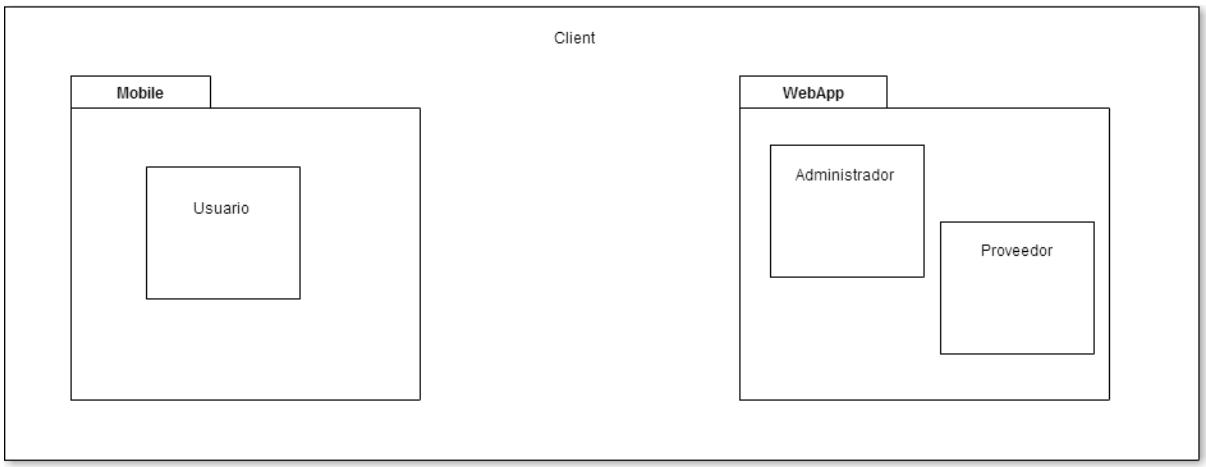
5. Logical View

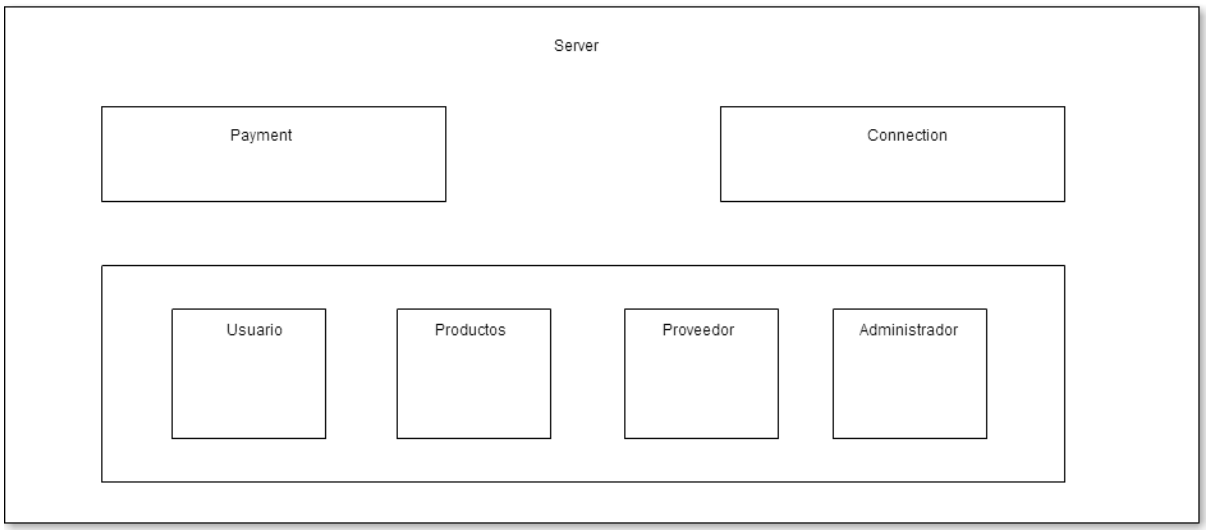
La parte crítica del proyecto es la transacción monetaria entre el usuario, el sistema y el proveedor.

Para esto tendremos un paquete llamado com.makeYourParty.payment, dentro de éste tendremos la clase pago que se conectará con el API de “Conekta”, que en este caso es el servicio que se utilizará para la realización de los pagos.

Otro servicio importante es la conexión de cada aplicación móvil con la base de datos a través de los servicios web. Para esto habrá un paquete llamado com.makeYourParty.connection y dentro de este paquete estará la clase Connection, que se conectará y consumirá los datos obtenidos del servicio web.

5.1 Overview

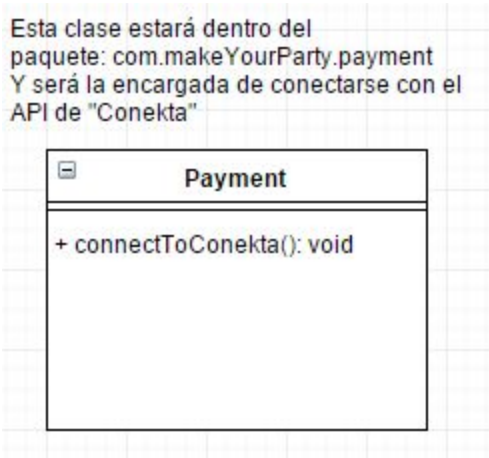




5.2 Architecturally Significant Design Packages

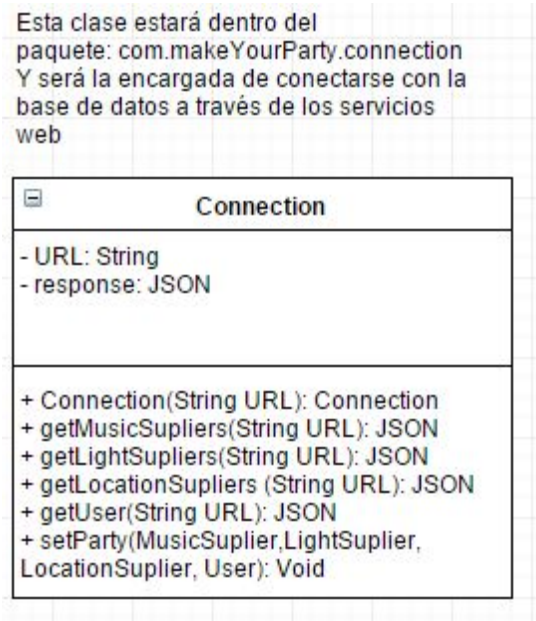
Payment: Obtención y consumo de datos crediticios del usuario.
Clases importantes:

- Payment: Conexión con el API de Conekta para las transacciones monetarias.

- 

Connection: Enlace entre las aplicaciones individuales (usuario, administrador, proveedor) con sus tablas correspondientes en la base de datos.
Clases importantes:

- Connection: Recibe los requests de las aplicaciones individuales, los clasifica y realiza las operaciones correspondientes en la base de datos (CRUD) dependiendo del tipo de petición.

- 

5.3 Use-Case Realizations

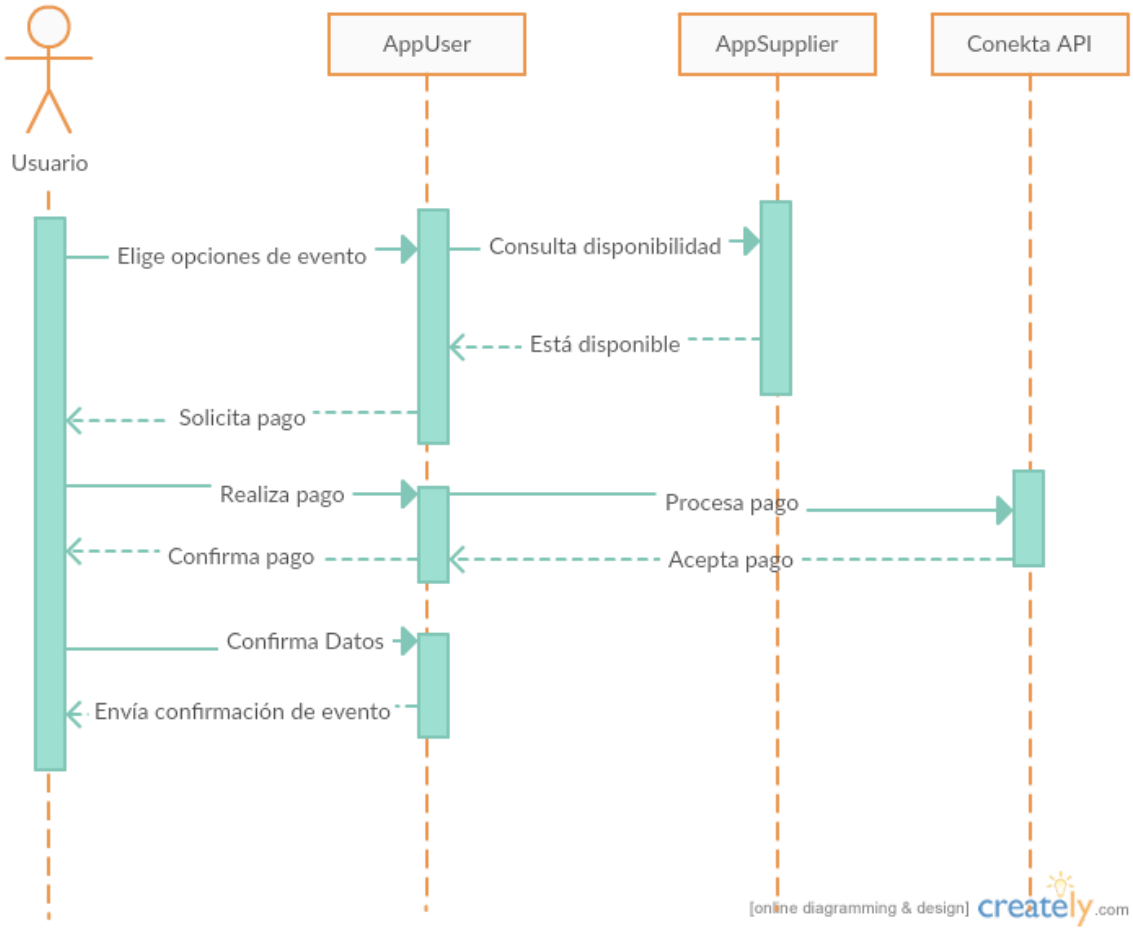
Registro:

El usuario introduce las credenciales que quiere que el sistema almacene. Se manda una petición al servidor de tipo “Registro”. La clase Conexión se dedica a hacer la operación de alta en la tabla de Usuario con las debidas propiedades de seguridad. Una vez hecho el cambio en la base de datos (o una vez detectado un error en la consulta), se le notifica al usuario si la operación fue exitosa o no.

Paga Fiesta:

Una vez seleccionados todos los productos, se envía una petición a Payment, que hace el enlace al módulo de Conekta (https://www.conekta.io/es/docs/referencias/conekta-android). Una vez hecho el pago (o una vez detectado un error en el proceso), se le notifica al usuario si la transacción fue exitosa o no.

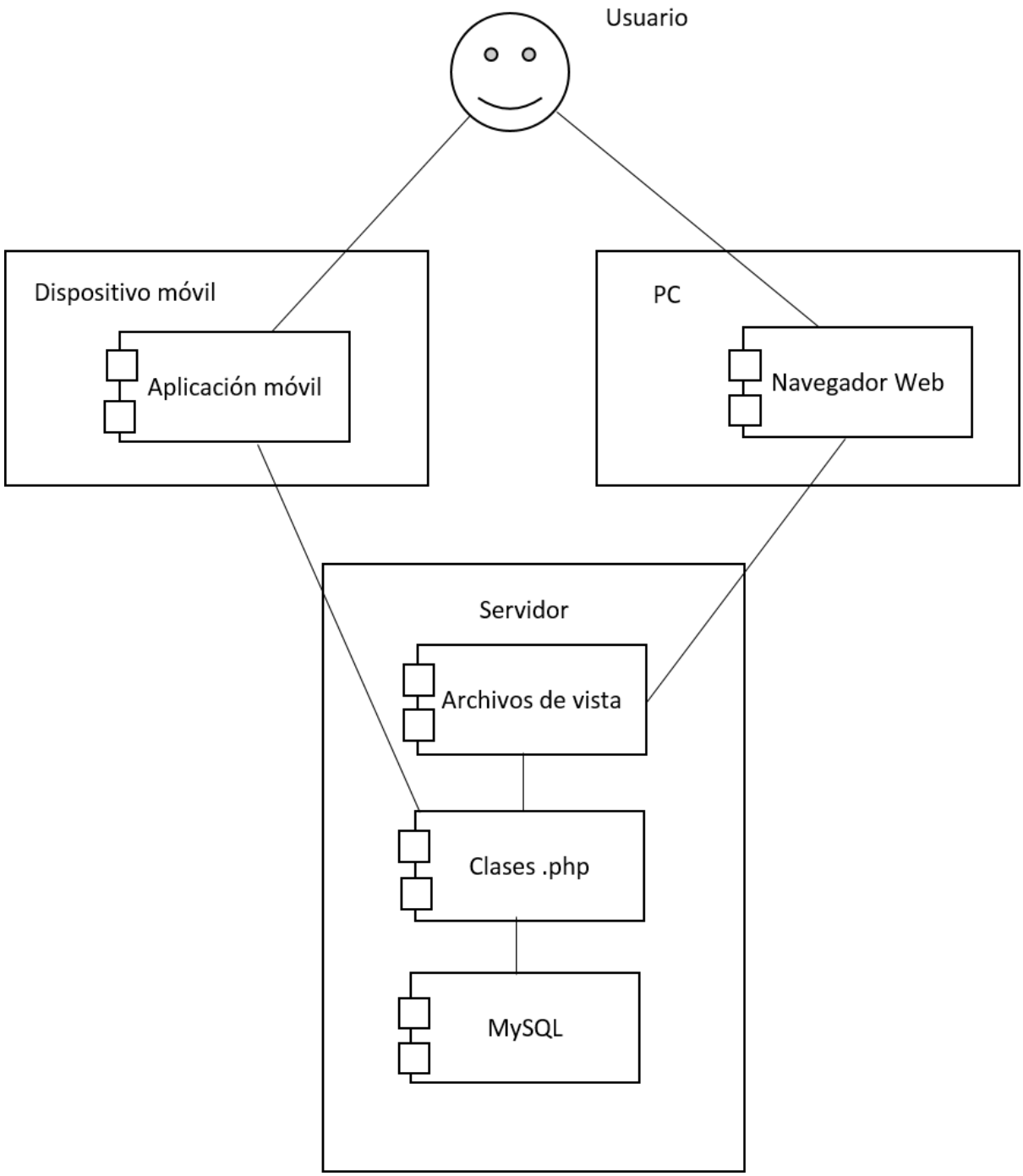
6. Process View



7. Deployment View

Los dispositivos en los que se desplegará el software son los siguientes:

- Dispositivos con sistema operativo Android
- Dispositivos con sistema operativo de iOS
- Dispositivos con navegadores web instalados
- Servidor central sobre internet



[This section describes one or more physical network (hardware) configurations on which the software is deployed and run. It is a view of the Deployment Model. At a minimum for each configuration it should indicate the physical nodes (computers, CPUs) that execute the software and their interconnections (bus, LAN, point-to-point, and so on.) Also include a mapping of the processes of the **Process View** onto the physical nodes.]

8. Implementation View

8.1 Overview

El estilo principal que utilizarán las tres aplicaciones será Cliente/Servidor. Se requiere de una plataforma centralizada que pueda responder a cada una de las peticiones de la aplicación móvil y las aplicaciones web. Se utilizará una base de datos centralizada como un repositorio. Sin embargo, el código encargado de las modificaciones sobre esta base de datos se encontrará del lado del servidor, por lo que la comunicación entre cliente y repositorio no será directa.

8.2 Layers

[For each layer, include a subsection with its name, an enumeration of the subsystems located in the layer, and a component diagram.]

9. Data View (optional)

[A description of the persistent data storage perspective of the system. This section is optional if there is little or no persistent data, or the translation between the Design Model and the Data Model is trivial.]

10. Size and Performance

En cuanto al tamaño estamos considerando que a largo plazo tendremos un aproximado de 100 proveedores y 10,000 usuarios mínimo. Esto nos da, como resultado aproximado, 20 Gb de almacenamiento en cuanto a base de datos. En cuanto a las aplicaciones móviles cada aplicación será de aproximadamente 20 Mb en disco.

En el performance de la aplicación lo que más impacta son las transacciones monetarias por lo que será necesario implementar librerías ya creadas que nos garanticen un alto nivel de desempeño en el proceso de transacciones.

11. Quality

Portabilidad
El sistema será desarrollado en android para versiones 4.1 en adelante , aproximadamente esto cubre el 94.8% de dispositivos android que están actualmente activos en Google Play Store.

Usabilidad/Experiencia de usuario

La interfaz de usuario de MakeYourParty estará diseñado para la facilidad de uso y deberá ser apropiada para una comunidad de usuarios con conocimientos básicos en aplicaciones de android sin formación adicional para poder utilizarla.

Seguridad
Los pagos serán realizados por la api de Conekta lo cual nos ofrece:

Encriptado:Cada bit de información transmitido es encriptado por medio de SSL en un flujo diseñado para garantizar la protección en el envío de todos los datos sensibles de cada transacción.

Estándares:Conekta cuenta con estándares de seguridad certificados en la industria de pagos para almacenamiento y procesamiento de transacciones con tarjeta, PCI DSS.

Tokenización de tarjetas: Cada tarjeta es encriptada automáticamente en los servidores de Conekta y es convertida en un token que puede ser almacenado de manera segura en nuestros servidores y ser utilizada para generar cargos a tarjetas.