República Bolivariana de Venezuela
Ministerio del Poder Popular para la Educación
Universidad de los Andes
Facultad de Ingeniería
Escuela de Sistemas

PLOP

Desarrolladores del Proyecto:

- Ana Guerrero CI 25.154.497
- Victor Rojas CI 23.391.502
- Gustavo Mejias CI 25.302.093

Justificación

Actualmente una necesidad de muchos negocios, comercios y establecimientos es hacer llegar información y promociones a su listado de clientes y a la mayor cantidad de personas posibles para mantener un negocio estable y con nuevos clientes. Aplicar estrategias de estrategias de mercado a través de publicidad visual ha tenido un gran impacto en las ventas y ganancias de muchos establecimientos comerciales.

El proyecto viene a suplir esta necesidad al ofrecer una nueva herramienta a los estudios de mercado y mercadeo, mediante un servicio de notificaciones push, en el cual un usuario con una marca sea capaz de difundir la información que desee a una lista de receptores, que él mismo provee, éstos acceden a recibir la información que provenga de dicho usuario, a cambio de estar informado de las marcas que le interesan, el receptor confía en su usuario que el uso de esta herramienta sea el debido.

El sistema contará con dos tipos de usuarios, el premium y básico, la diferencia de estos radica en el hecho de que el usuario premium tendrá ciertas características como editar y reenviar las notificaciones. El pago se realizará en línea dentro del software PLOP.

El objetivo del proyecto es contar con dos entornos: uno web para el usuario que sea capaz de mandar notificaciones y uno móvil que sea capaz de recibirlas, de esta manera ayudar a los comercios a mejorar sus ingresos mensuales e incrementar su listado de clientes.

Cliente:

Nombre: Edgar Ramón Pérez Godoy.
C.I. 26.907.745
Ocupación: Desarrollador web.

AplicaciónWeb: Envio de notificaciones push

Especificación: Sitio Web para enviar notificaciones push.

LISTA DE REQUERIMIENTOS APLICACIÓN WEB

Id	Nombre	Descripción	Tipo
1	Envío de notificaciones push	Pagina web que permita el envió de notificaciones push a uno o varios receptores.	Funcional
2	Sistema de registro de usuarios	Dos tipos de usuario: Emisor y Receptor.	Funcional
3	Usuario Enviador	Usuario que enviará notificaciones a usuarios receptores.	Funcional
4	Usuario Receptor	Usuario que recibirá notificaciones de usuarios emisores.	Funcional
5	Versiones de la página	Dos versiones: Básica y premium.	Funcional
6	Versión Premium	Creación de grupos, notificaciones ilimitadas, reenvío de notificaciones, opciones de personalización.	Funcional
7	Versión básica	Notificaciones limitadas. Sin posibilidad de crear grupos y sin reenvió de notificaciones.	Funcional
8	Sistema de pago	Por tarjeta de crédito con una cuota única.	Funcional
9	Diseño	Diseño basado en material design, ligero, de fácil uso.	No Funcional
10	Paleta de colores	Tonos turquesa con combinación de colores pasteles.	No Funcional

Aplicación Móvil: Receptor de notificaciones push

Especificación: Aplicación móvil para recibir notificaciones.

LISTA DE REQUERIMIENTOS APLICACIÓN MÓVIL

Id	Nombre	Descripción	Tipo
11	Recepción de notificaciones push	Aplicación móvil que permita a los usuarios receptores recibir notificaciones push de uno o varios usuarios emisores.	Funcional
12	Usuario: Receptor	Usuario que recibirá notificaciones de usuarios enviadores. A la aplicación móvil PLOP solo podrán acceder los usuarios de tipo Receptor.	Funcional
13	Diseño	Diseño basado en material design, ligero, de fácil uso.	No Funcional
14	Paleta de colores	Tonos turquesa con combinación de colores pasteles.	No Funcional
15	Plataforma	Disponible para dispositivos android	No Funcional

MATRIZ DE TRAZABILIDAD ENTRE LA DEPENDENCIA DE REQUISITOS

Id	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1														
2														
3		X												
4		X												
5														
6			X		X									
7				X	X									
8			X			X								
9														
10										X				
11	X													
12		X		X										
13									X					
14										X				
15														

ARQUITECTURA DEL PROYECTO

Luego de estudiar los sistemas y métodos de trabajo junto con el equipo de trabajo y el cliente, y al obtener conocimiento del modelo de negocio, se decidió utilizar **React** como framework de trabajo de la aplicación PLOP.

El framework React o React js es una librería de Javascript de código abierto para crear interfaces de usuario con el objetivo de animar el desarrollo de aplicaciones de una sola pagina. Con React es posible construir aplicaciones que usan datos que cambian constantemente, por esto es fácil de combinar. Además es una excelente opción para aplicaciones que necesitan manejar una gran cantidad de datos, ya que ofrece grandes beneficios en performarnce, modularidad y promueve un flujo muy claro de datos y eventos, facilitando la planeación y desarrollo de apps complejas.

Para el desarrollo de la aplicación se planteó establecer una arquitectura **cliente / servidor** desacoplada. Es decir, una parte front-end (Vista) donde se utilizaran tecnologías como javascript (React) mas soluciones CSS como BootStrap de tal forma que el back-end (Controller + Modelo) proporcionarse un conjunto de servicios REST que se comunican mediante el uso del JSON.

Al utilizar esta arquitectura se consigue que la transmisión de datos entre el cliente y el servidor sea la estrictamente necesaria, ya que los datos transferidos son representados como objetos JSON que son fácilmente interpretados en la parte cliente.

En cuanto al Back-End, se usará el lenguaje de programación Go (Golang). Go es un lenguaje desarrollado por Google y su principal ventaja es que no necesita un sistema operativo especifico para su uso, esta característica es llamada cross compilation. Otra de sus grandes ventajas es su simplicidad así como su velocidad y rendimiento.

METODOLOGÍA

La metodología a utilizar es la de modelos ágiles, ya que principalmente contamos con un cliente involucrado completamente con el proyecto, el cual desea una entrega incremental del mismo, estas dos características pertenecen a los modelos ágiles. De las técnicas de modelos ágiles se escogió la técnica de scrum ya que se adapta a la forma de trabajo que el cliente requiere, además de esto tiene

bondades para que el equipo de trabajo funcione correctamente como las reuniones de avance diaremente, todo el equipo de trabajo debe estar al día con el conocimiento de los avances y problemas del proyecto.

PATRONES DE DISEÑO

Template Method: El patrón de método de la plantilla es un patrón de diseño de comportamiento que define el esqueleto de programa de un algoritmo en un método, llamado método de plantilla, el cual difiere algunos pasos a las subclases.

El método de plantilla está diseñado para marcos, donde cada cual implementa las partes invariables de la arquitectura de un ámbito, dejando "placeholders" para personalizar las opciones. Esto es un ejemplo de inversión de control. Algunas razones por la que se utiliza el método de plantilla son:

- Dejar que las subclases que se implementan (a través del método primordial) tengan un comportamiento que puede variar.

- Evitar duplicación en el código: la estructura general de flujo de trabajo, está implementada una vez en el algoritmo de clase abstracta, y variaciones necesarias son implementadas en cada de las subclases.

- Control en qué punto(s) la subclassing está permitida. En oposición a una sencilla sobrecarga polimórfica, donde el método de base sería enteramente reescrito, permitiendo un cambio radical en el flujo. Sólo los detalles específicos del flujo se pueden cambiar.

Este tipo de patrón se usa en lo que es llamado en el código de nuestro proyecto componentes en la parte de react.js y react native.

Observer: Es un patrón de diseño de software que define una dependencia del tipo uno a muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes.

Este patrón suele utilizarse en los entornos de trabajo de interfaces gráficas orientados a objetos, en los que la forma de capturar los eventos es suscribir listeners a los objetos que pueden disparar eventos.

State: Se utiliza cuando el comportamiento de un objeto cambia dependiendo del estado del mismo. En determinadas ocasiones, cuando el contexto en el que se está desarrollando requiere que un objeto tenga diferentes comportamientos según el estado en que se encuentra, resulta complicado poder manejar el cambio de comportamientos y los estados de dicho objeto, todos dentro del mismo bloque de código. El patrón State propone una solución a esta complicación, creando básicamente, un objeto por cada estado posible del objeto que lo llama.

ESPECIFICACIÓN DE REQUISITOS

1. **Envió de notificaciones push:** un usuario debe ser capaz de enviar notificaciones a el teléfono de un receptor.

1.1 El usuario desde una pagina web debe ser capaz de entrar a un apartado donde contenga un formulario con los datos de la notificación.

1.2 Dentro del formulario debe estar varios tipos de notificaciones, entre los tipos debe estar una simple(titulo, cuerpo), con botón(ademas de titulo y cuerpo, debe poseer un botón para ir a una url), con imagen(una simple que posee una imagen en la notificación).

1.3 Una vez el usuario seleccione la notificación que desea enviar, el formulario debe cambiar de acuerdo a los parámetros requeridos para cada tipo de notificación.

1.4 Luego debe aparecer un apartado para elegir los receptores de dicha notificación, debe poseer un buscador y poder segmentarse.

2. **Tipos de usuario:** Debe existir la posibilidad de que los usuarios se registren, un usuario debe tener una lista de receptores, y estos pueden estar suscritos a varios usuarios.

2.1 Registro donde especifique que tipo de usuario va a ser, deben aparecer las dos opciones de manera atractiva, luego pedir los datos correspondientes para el registro.

3. **Usuario emisor:** Debe existir roles entre los usuarios, para poder separarlos.

3.1 Uno es el rol free, que debe estar restringido con solo la posibilidad de enviar notificaciones simples.

3.2 Otro de los roles es el roll premium, el cual debe tener todas las opciones disponibles para el envio de notificaciones.

4. **Usuario receptor:** este usuario debe tener algunos datos de segmentación, como país, lenguaje, edad.

4.1 En el registro debe aparecer el logo de la aplicación si este usuario se esta registrando a un usuario free, en caso contrario debe mostrarse el logo del usuario al que se esta suscribiendo.

4.2 Debe desplegarse un formulario con todos los datos necesarios para el registro; una vez realizado el registro se debe indicar el link desde el cual puede obtener la aplicación.

5. **Dos versiones de la web:** El usuario que es free posee opciones limitadas en el uso de la plataforma, cuando intente acceder a alguna opción a la cual no tiene acceso debe aparecer un mensaje con publicidad para hacerse premium.

6. **Creación de grupos, notificaciones ilimitadas, reenvío de notificaciones, opciones de personalización:** Estas opciones estarán disponibles para aquellos usuarios que son premium.

6.1 Los grupos deben poseer nombre, una vez elegido, se selecciona que usuarios receptores pertenecen a ella.

6.2 Las notificaciones no deben de poseer limites, es decir que se pueden crear y reenviar cuantas veces quiera el usuario.

6.3 El usuario debe ser capaz de modificar su logo, además de otras opciones como modificar la imagen de fondo que le aparece a los usuarios receptores que se van a suscribir a él.

7. **Notificaciones limitadas**. Para aquellos usuarios free. Sin posibilidad de crear grupos y sin reenvió de notificaciones.

8. **Sistema de pago:** Opción disponible para cambiar un usuario de rol free a premium a través de una plataforma de pagos.

8.1 Cuando el usuario quiera pasar a premium debe aparecer un formulario para introducir todos los datos de la tarjeta de crédito desde la cual va a realizar el pago.

8.2 La plataforma de pago puede ser una de estas: mercadopago, payu, paypal(en caso de salga al exterior).

11. **Recepción de notificaciones push:** Aplicación móvil que permita a los usuarios receptores recibir notificaciones push de uno o varios usuarios emisores.

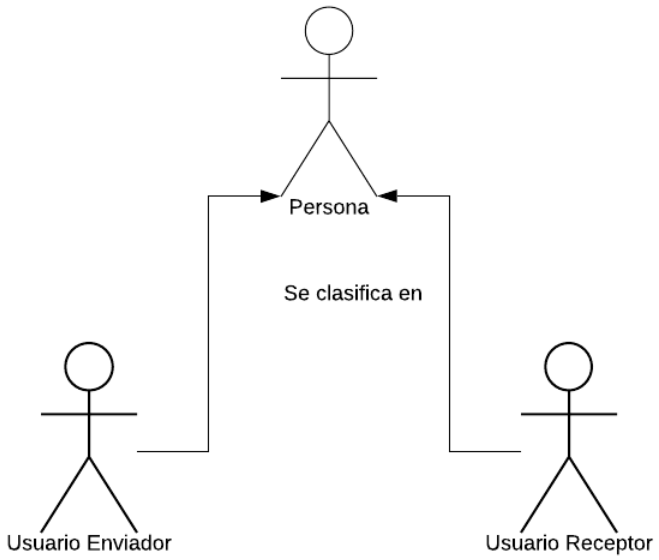
11.1 Cuando el usuario se registre y posea la app, esta deberá tener una lista de los usuarios a los que pertenece.

11.2 Cuando se presiona sobre el logo de un usuario, deben aparecer las notificaciones recibidas desde ese usuario.

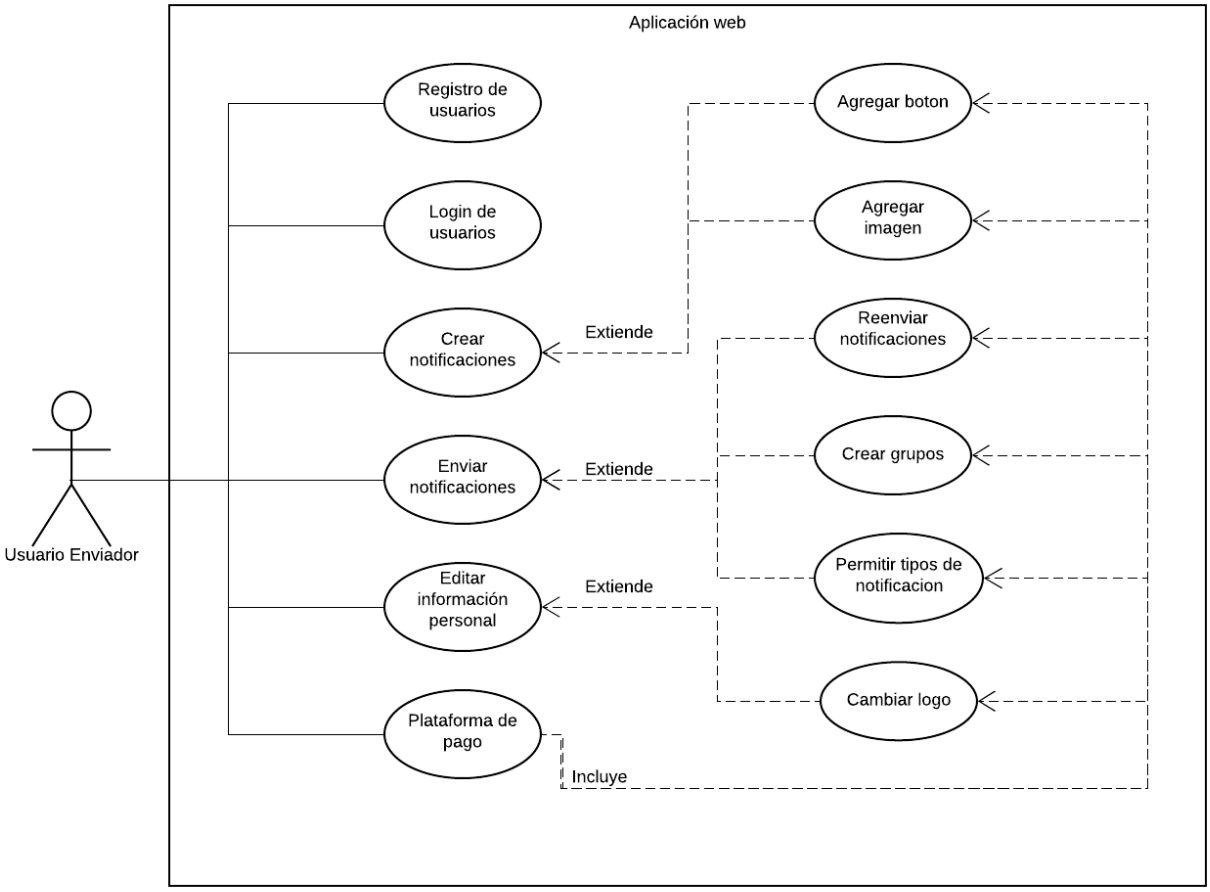
12. **Usuario:** Receptor, es aquel usuario que recibirá notificaciones de los usuarios emisores.

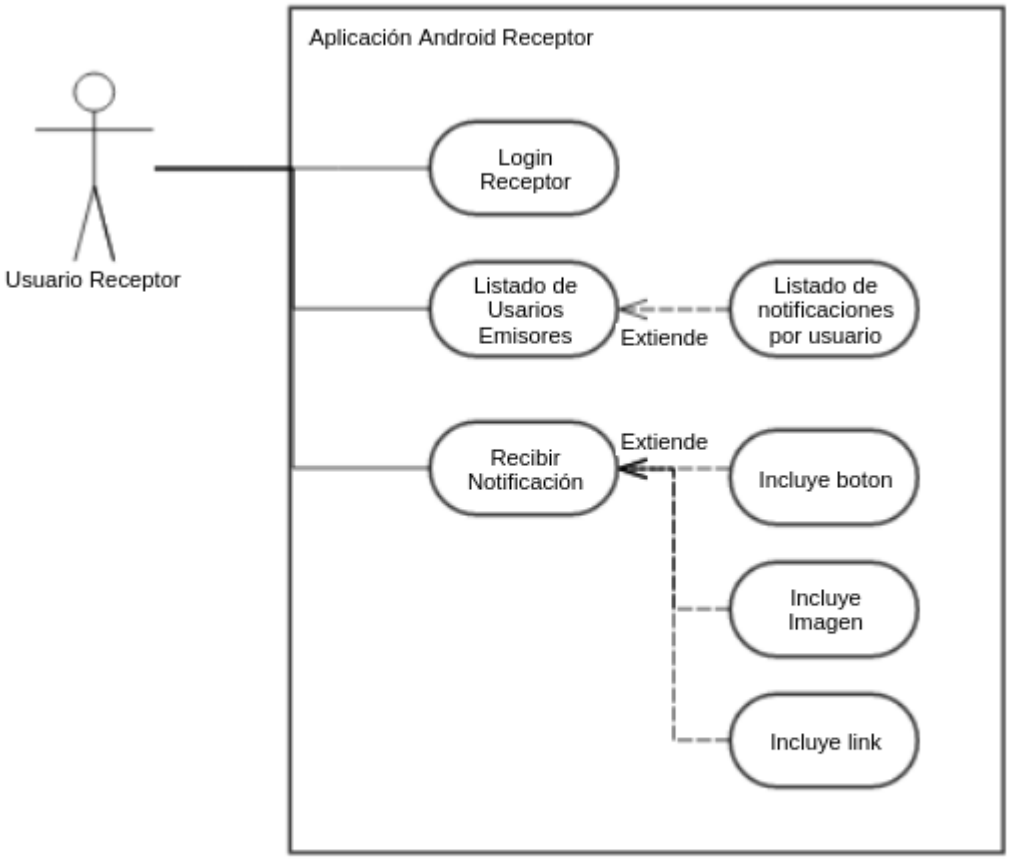
12.1 Registro muy simple en la app para ligar al receptor que se registro en la web.

MODELO DE ACTORES



CASOS DE USO


Aplicación web
Registro de usuarios
Login de usuarios
Crear notificaciones
Enviar notificaciones
Editar información personal
Plataforma de pago
Usuario Enviador
Extiende
Extiende
Extiende
Incluye
Agregar boton
Agregar imagen
Reenviar notificaciones
Crear grupos
Permitir tipos de notificacion
Cambiar logo


Aplicación Android Receptor
Usuario Receptor
Login Receptor
Listado de Usarios Emisores
Listado de notificaciones por usuario
Extiende
Recibir Notificación
Extiende
Incluye boton
Incluye Imagen
Incluye link

TRANSFORMACIÓN ERE RELACIONAL

Features (Idfeatures, Send_Notification, Resend_notification, Edit_Notification, Create_Groups, Custom_Logo, Notification_Types, Schedule_Notification)
Listerners (CI)
Devices (Token, End_point, P256h, Auth, Phonenumber, Os, *CI_Listeners*)
Listeners_Receive_Notifications (Idnotifications, CI_Listeners)
Listeners_Has_Users (CI_Users, CI_Listeners)
Users_Send_Notifications (Idnotifications, CI_Users)
Notifications (Idnotifications, Title, Body, Src_image, Name_Button, Action, Type)
People (CI, Countries, Gender, City, Name, Src_logo, Src_icon, Zip_code)
Rols (Idrol, Role_name, *Features_Idfeatures*)
Users (CI, Password, Email, Created_at, Updated_at, *Rols_Idrols*)

NORMALIZACIÓN A 3FN

- Tablas:
Notifications, Users, Listeners, Rols, Features, Devices, Listeners_Receive_Notifications, Listeners_Has_Users, Users_Send_Notifications
1FN
No existen relaciones anidadas, ni valores multivaluados. Todos sus atributos son atómicos.
2FN
Ningún atributo depende funcionalmente de parte de la clave
3FN
Ningún atributo que no pertenece a la clave depende funcionalmente de otro atributo que no es clave.

- Tabla People
1FN
Los atributos City y Country pueden ser tomados como atributos no atómicos.

1FN
People (CI, Gender, Name, *Countries_Idcountry*)
Countries (Idcountries, Idcities, name_country, name_city, zip_code)

2FN
No esta en 2FN ya que Idcities → name_city, zip_code

People (CI, Gender, Name, *Countries_Idcountry*)
Cities (Idcities, name, zip_code, *Countries_Idcountry*)
Countries (Idcountries, name)

3FN
Ya se encuentra en 3FN

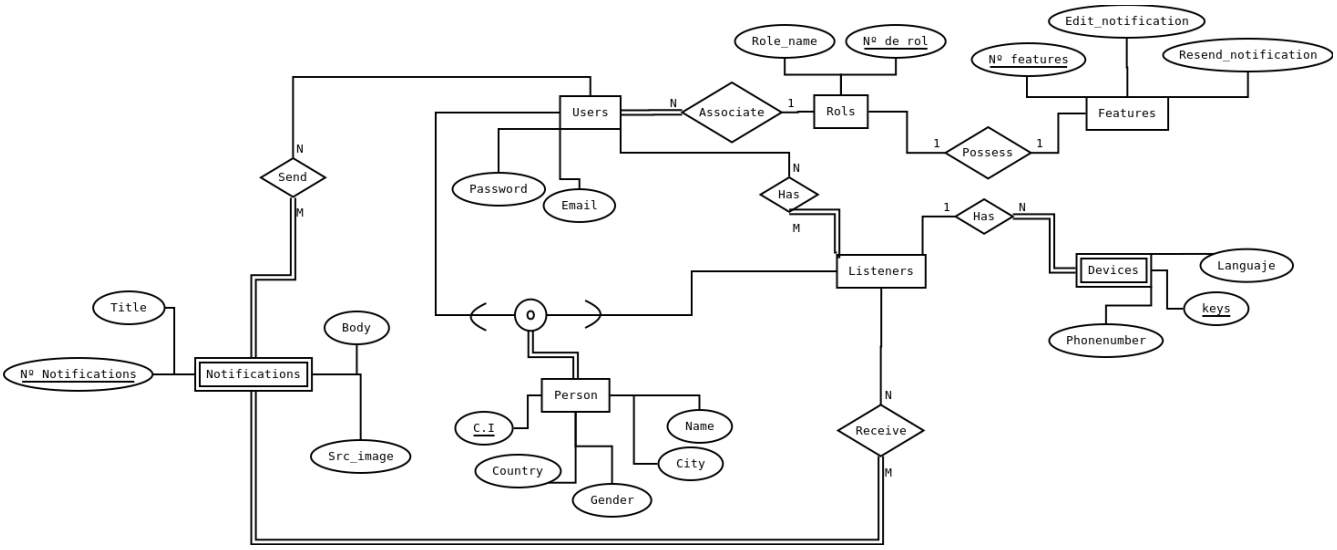
Explicación atributos
- Resend_Notification: Se usará para determinar si un usuario puede reenviar notificaciones.
- Edit_Notification: Se usará para determinar si un usuario puede editar notificaciones.

- Create_Groups: Se usará para determinar si un usuario puede crear grupos.
- Custom_Logo: Se usará para determinar si un usuario puede modificar la foto de perfil
- Notification_Types: Se usará para determinar si un usuario puede enviar distintos tipos de notificaciones.

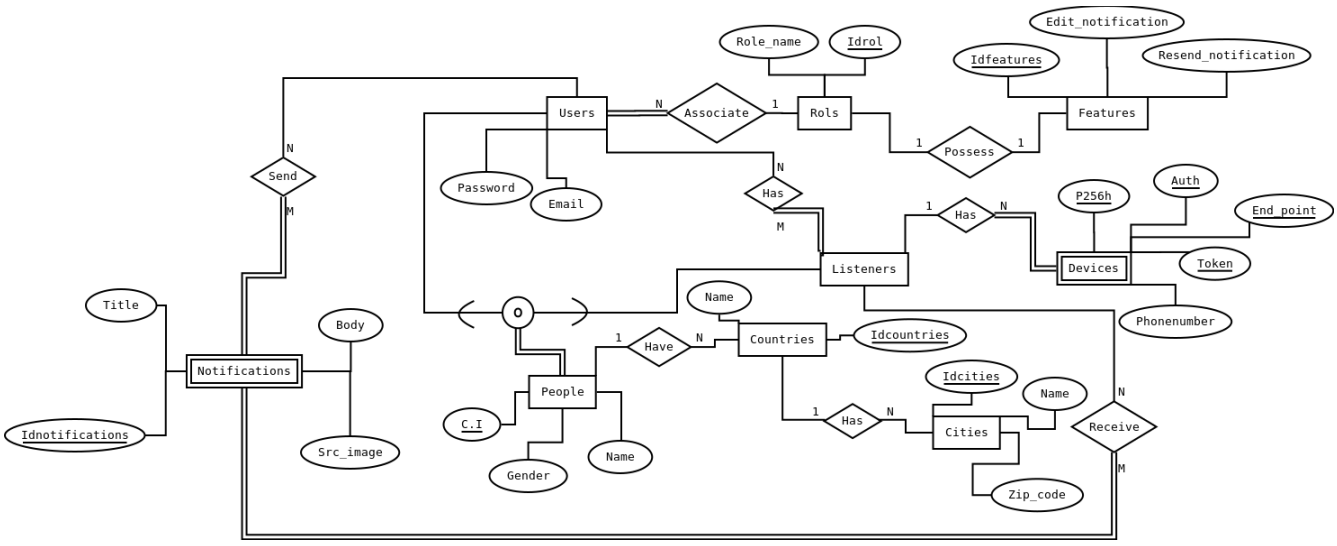
CAMBIOS EN LA ESTRUCTURA DEL PROYECTO

ANTES

Modelo ERE sin normalizar

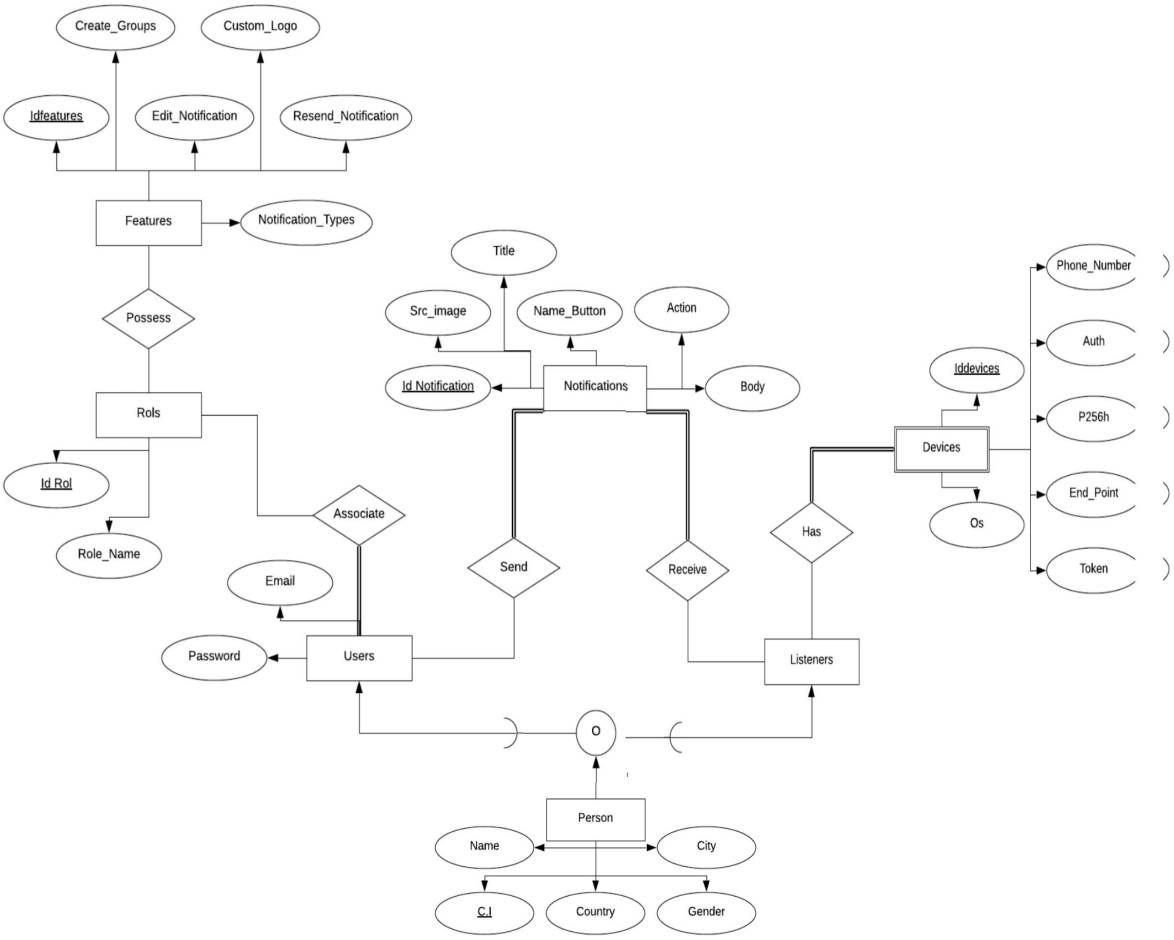


Modelo ERE normalizado

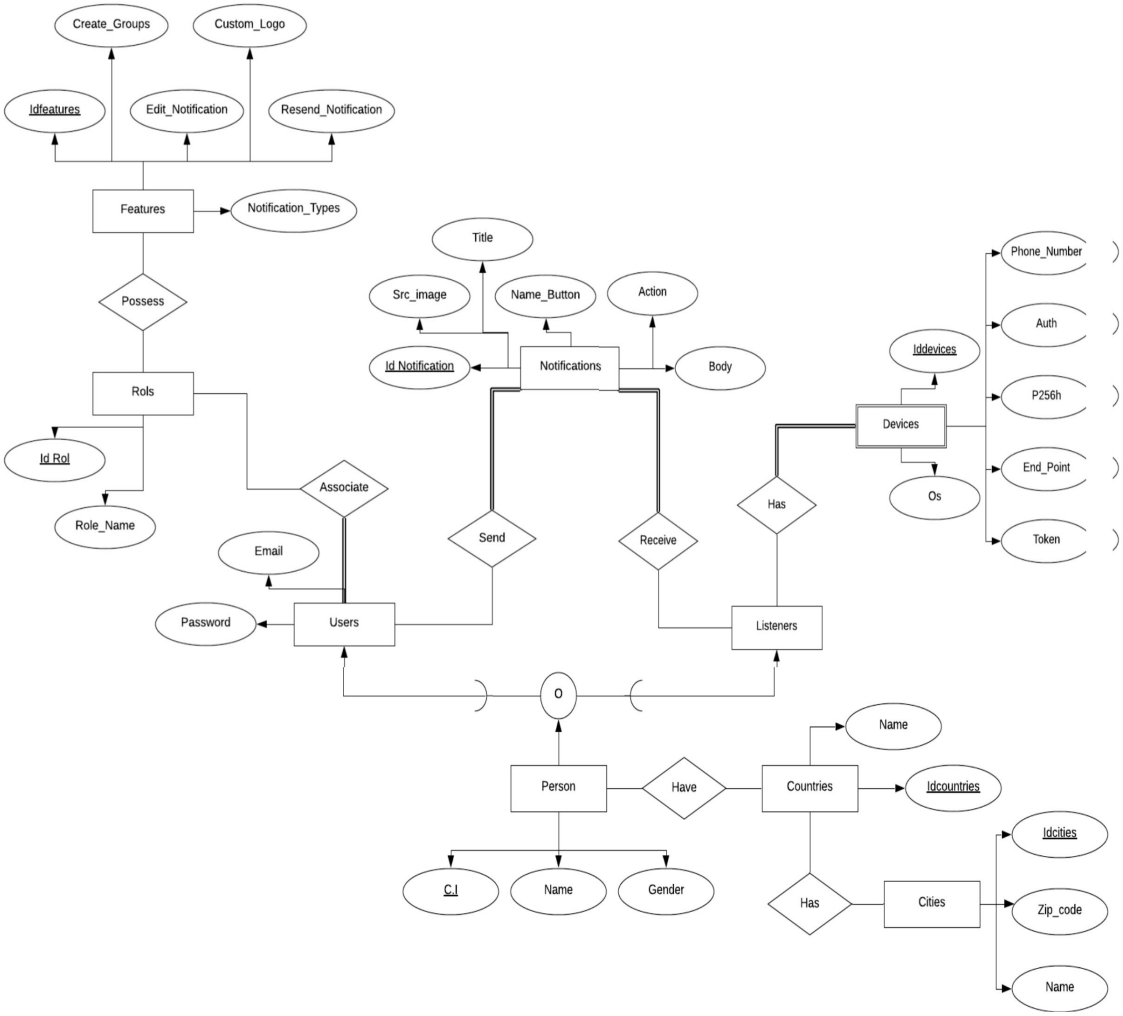


DESPUÉS

Modelo ERE sin normalizar

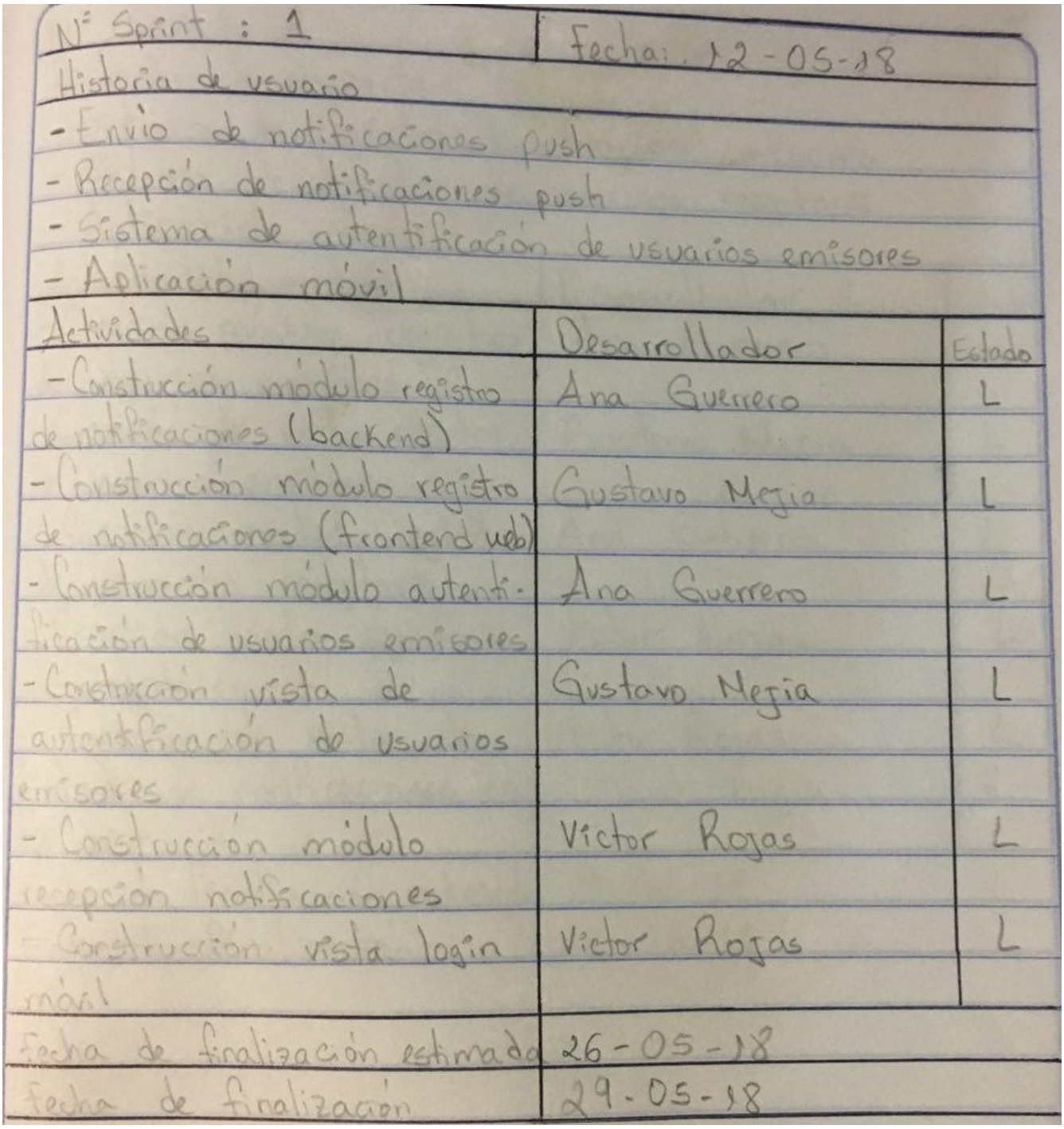


Modelo ERE normalizado



SPRINTS REALIZADOS

Sprint #1:



N° Sprint : 1	Fecha: 12-05-18	
Historia de usuario		
- Envío de notificaciones push		
- Recepción de notificaciones push		
- Sistema de autentificación de usuarios emisores		
- Aplicación móvil		
Actividades	Desarrollador	Estado
- Construcción módulo registro de notificaciones (backend)	Ana Guerrero	L
- Construcción módulo registro de notificaciones (fronterd web)	Gustavo Mejia	L
- Construcción módulo autenti-ficación de usuarios emisores	Ana Guerrero	L
- Construcción vista de autentificación de usuarios emisores	Gustavo Mejia	L
- Construcción módulo recepción notificaciones	Victor Rojas	L
- Construcción vista login móvil	Victor Rojas	L
Fecha de finalización estimada	26-05-18	
Fecha de finalización	29-05-18	

Sprint #2:

N° Sprint: 2	Fecha: 30-05-18	
Historia de usuario: - Sistema de registro de usuarios emisores - Sistema de registro de usuarios receptores - Aplicación móvil		
Actividades	Desarrollador	Estado
- Creación módulo registro de usuarios emisores	Ana Guerrero	L
- Creación de vista registro de usuarios emisores	Gustavo Mejia	L
- Creación modulo registro de usuarios receptores	Ana Guerrero	L
- Creación vista registro de receptores	Victor Rojas	L
- Creación de vista de listado de notificaciones en la app	Victor Rojas	L
Fecha de finalización estimada	11-06-18	
Fecha de finalización	16-06-18	

Sprint #3:

N° de Sprint: 3	Fecha: 17-06-18	
Historia de usuario - Reenvio de notificaciones - Suscripción usuario emisor - Creación de grupos - Sistema de pago		
Actividades	Desarrollador	Estado
-Creación modulo reenvio de notificaciones.	Ana Guerrero	P
-Creación vista reenvio de notificaciones	Gustavo Mejia	P
-Creación vista de grupos	Gustavo Mejia	L
- Implementación sistema de pago	Victor Rojas	EP
-Creacion modulo de pago	Ana Guerrero	L
- Creación vista modulo de pago	Victor Rojas	EP
fecha de finalización estimada	25-06-18	
fecha de finalización	27-06-18	