UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS

22-4-2015

TRABAJO 02

Aplicación Móvil: RestaurApp

**ASIGNATURA:** CC11 Desarrollo para Dispositivos Móviles

**PERÍODO ACADÉMICO:** 2015-I

**INTEGRANTES:**

* Baquerizo Nuñez, Andrés
* Canales Zorrilla, Rodrigo
* Sandoval Nuñovero, José
* Taza Rojas, Roy

1. **DESCRIPCIÓN DEL PROBLEMA A RESOLVER**

El problema que buscamos resolver es la ausencia de una forma para recomendar y compartir rápidamente los gustos particulares de un restaurante con otras personas manteniendo la información importante y a la vez concisa.

Por lo que se propone crear una aplicación móvil que permita registrar la información de los restaurantes de la cuidad para agregar fotografías, comentarios y la ubicación GPS, con la finalidad de compartirla con otros usuarios, los cuales pueden calificar y volver a recomendar a otros usuarios.

1. **DEFINICIÓN DEL ALCANCE DEL PROYECTO**
   1. **Limitar el alcance del proyecto**

Como alcance inicial del proyecto tenemos, que existirán 2 formas de acceso a la aplicación, registrándose como nuevo usuario indicando los datos principales o usar el API de Facebook para iniciar sesión con la cuenta personal. También se podrá sincronizar una cuenta de Facebook con una cuenta creada en la aplicación.

La aplicación tendrá una lista de restaurantes las cuales podrá será filtrada automáticamente de acuerdo a preferencias del usuario, las cuales serán ingresadas al momento de registrarse o en el perfil de éste. También tendrá la funcionalidad de hacer búsquedas de recomendaciones hechas por otros usuarios.

Para el registro de una recomendación, el usuario puede agregar fotografías, comentarios, ubicación GPS, poner una calificación, puede elegir la visibilidad de la recomendación (pública, dirigida a un usuario o grupo de contactos) y por último puede postear la recomendación en Facebook.

* 1. **Requerimientos**

**2.2.1 Requerimiento Funcional 01 (RF01):** El sistema debe permitir iniciar sesión por una cuenta previamente creada por el usuario o por el API de Facebook.

**2.2.2 Requerimiento Funcional 02 (RF02):** El sistema debe permitir buscar los restaurantes más recomendados, de acuerdo a preferencias del usuario.

**2.2.3 Requerimiento Funcional 03 (RF03):** El sistema debe permitir hacer búsquedas de recomendaciones por distrito.

**2.2.4 Requerimiento Funcional 04 (RF04):** El sistema debe poder registrar una recomendación adjuntando a ella comentarios, fotografías, ubicación GPS (si aún no existe restaurante) y calificación.

**2.2.5 Requerimiento Funcional 05 (RF05):** El sistema debe permitir el envío de recomendaciones a un usuario o grupo de contactos.

**2.2.6 Requerimiento Funcional 06 (RF06):** El sistema debe permitir postear una recomendación en Facebook.

* 1. **Objetivos**

**2.3.1 Objetivo Principal**

El grupo pretende desarrollar una aplicación de calidad que brinde al usuario una ayuda al momento de buscar un lugar para comer de acuerdo a sus preferencias y recomendaciones del público en general o de sus contactos. Para lo cual se quiere integrar dicha aplicación con las redes sociales para hacerla mucho más amigable, en el sentido que puedas tener opiniones de amigos cercanos. Así como también se busca aplicar el concepto de geolocalización para que el usuario tenga un panorama mucho más eficaz sin tener que estresarse buscando por su propia cuenta lugares cercanos donde poder comer.

**2.3.2 Objetivos Específicos**

1. El sistema deberá ser de fácil acceso debido a la integración con el API de Facebook, lo cual permitirá que muchas más personas usen la aplicación sin tener que hacer toda la travesía de registrarse.
2. El usuario podrá encontrar los restaurantes más recomendados basados en sus preferencias personales de categoría.
3. El sistema facilitará al usuario la ubicación de los restaurantes más recomendados cercanos a él.
4. El usuario podrá realizar una recomendación a un restaurante que le haya parecido agradable usando comentarios fotos y calificaciones
5. El usuario podrá recomendar un restaurante a un amigo, así como también recibir recomendaciones de sus contactos para poder tener una mejor decisión al momento de ir a consumir a un restaurante.
6. El usuario podrá publicar sus recomendaciones en Facebook para que más personas puedan conocerla.
7. **DEFINICIÓN DE LA ARQUITECTURA DEL PROYECTO**
   1. **Módulos**

Los módulos con los que contará nuestra aplicación son los siguientes:

* + 1. **Módulo de Usuarios:**

Es el que contiene las funcionalidades de perfil de usuarios, gustos y preferencias por las diferentes categorías de comida.

* + 1. **Módulo de Recomendaciones:**

En este módulo se agrupan las funciones más importantes de la aplicación. Se cubren las funciones de tomar y subir fotos relacionadas a la recomendación de un restaurante, comentar acerca de un restaurante, utilizar geolocalización para ubicar el restauramos en el que nos encontramos y queremos recomendar. Además hemos considerado la posibilidad de poder publicar nuestra recomendación en nuestro perfil de Facebook.

* + 1. **Módulo de Redes Sociales:**

Gracias a este módulo podremos permitir a nuestros usuarios iniciar sesión en nuestra aplicación con sus cuentas de Facebook. Además, Este módulo llevará el control de la sincronización de cuentas entre las creadas con Facebook y las creadas normalmente, diferenciándolas o relacionándolas con los usuarios respectivos.

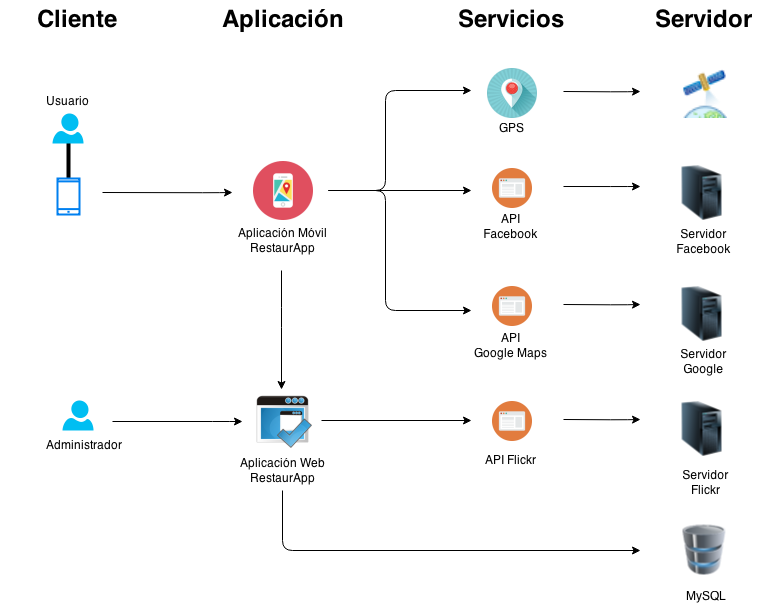
* + 1. **Módulo de Administrador Web:**

Este módulo se encarga del mantenimiento de los datos de los restaurantes, como sus datos, categorías, etc.

* 1. **Tecnologías**

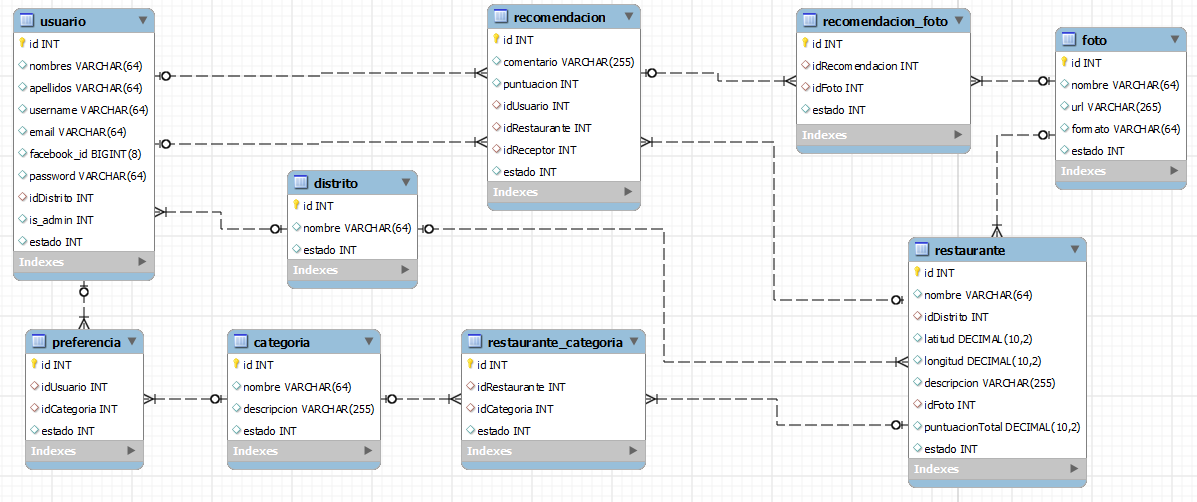
Las Tecnologías que hemos considerado aplicar son las siguientes:

1. Versión de Android: 4.4
2. Framework para Android: Android Studio 1.1
3. Permisos: Cámara, Conexión a internet
4. Api de Facebook 4.0.1
5. Base de Datos: MySql
6. Servidor para alojamiento de las fotos: Flickr
7. **DIAGRAMA DE ARQUITECTURA PROPUESTO**



1. **MODELO DE BASE DE DATOS**

A continuación, se presenta el diagrama de la base de datos RestaurApp. Se muestran las tablas que almacenarán la data necesaria para el funcionamiento correcto de la aplicación.



**Diccionario de Datos**

La siguiente tabla presenta el diccionario de datos de la aplicación. En ella se muestran los campos de la base de datos agrupados por entidad (con su respectiva descripción). Asimismo, se indica que atributos son Primary Key(PK) ó Foreign Key (FK), así como la siguiente información: Tipo(tipo de datos), Default(valor por defecto) y descripción.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Entidad** | **Descripción entidad** | **Llave** | **Campo** | **Tipo** | **Default** | **Descripción** |
| **usuario** | Entidad que guarda la información relacionada con el usuario | **PK** | id | int |  | Id del usuario |
|  | nombres | varchar(64) |  | Nombres del usuario |
|  | apellidos | varchar(64) |  | Apllidos del usuario |
|  | username | varchar(64) |  | Nombre de usuario |
|  | email | varchar(64) |  | Dirección de correo del usuario |
|  | facebook\_id | bigint(8) |  | Id de la cuenta de facebook relacionada con el usuario |
|  | password | varchar(64) |  | Contraseña del usuario |
| FK | idDistrito | int |  | Id del distrito del usuario |
|  | is\_admin | int | 0 | Indica si el usuario es o no administrador |
|  | estado | int | 1 | Indica si el usuario está activo o inactivo |
| **restaurante** | Entidad que guarda la información relacionada con un restaurante | **PK** | id | int |  | Id del restaurante |
|  | nombre | varchar(64) |  | Nombre del restaurante |
| FK | idDistrito | int |  | Id del distrito del restaurante |
|  | latitud | decimal(10, 2) |  | Latitud en que se encuentra el restaurante |
|  | longitud | decimal(10, 2) |  | Longitud en que se encuentra el restaurante |
|  | descripcion | varchar(255) |  | Descripción del restaurante |
| FK | idFoto | int |  | Id de la foto del restaurante |
|  | puntuacionTotal | decimal(10, 2) | 0 | Puntuación total del restaurante |
|  | estado | int | 1 | Indica si el restaurante está activo o inactivo |
| **foto** | Entidad que guarda la información de una foto. | **PK** | id | int |  | Id de la foto |
|  | nombre | varchar(64) |  | Nombre de la foto |
|  | url | varchar(255) |  | Url de la foto |
|  | formato | varchar(64) |  | Formato de la foto |
|  | estado | int | 1 | Indica si la foto está activo o inactiva |
| **distrito** | Entidad que guarda los distritos. | **PK** | id | int |  | Id del distrito |
|  | nombre | varchar(64) |  | Nombre del distrito |
|  | estado | int | 1 | Indica si el distrito está activo o inactivo |
| **recomendacion** | Entidad que guarda una recomendación. Asimismo, esta entidad se puede utilizar para recomendar a otro usuario. | **PK** | id | int |  | Id de la recomendación |
|  | comentario | varchar(255) |  | Comentario del usuario sobre el restaurante |
|  | puntuacion | int |  | Puntuación del restaurante dada por el usuario |
| FK | idUsuario | int |  | Id del usuario que realizó la recomendación |
| FK | idRestaurante | int |  | Id del restaurante relacionado con la recomendación |
| FK | idReceptor | int |  | Id del usuario al que otro usuario recomendó un restaurante |
|  | estado | int | 1 | Indica si la recomendación está activa o inactiva |
| **recomendacion\_foto** | Entidad que guarda las fotos relacionadas con una recomendación. | **PK** | id | int |  | Id de recomendación\_foto |
| FK | idRecomendacion | int |  | Id de la recomendación |
| FK | idFoto | int |  | Id de la foto relacionada con la recomendación |
|  | estado | int | 1 | Indica si recomendacion\_foto está activo o inactivo |
| **categoria** | Entidad que guarda los tipos de comida que sirven los restaurantes. | **PK** | id | int |  | Id de la categoría |
|  | nombre | varchar(64) |  | Nombre de la categoría. Indica si es Comida marina, criolla, etc. |
|  | descripcion | varchar(255) |  | Descripcion de la categoría de restaurante |
|  | estado | int | 1 | Indica si la categoría está activa o inactiva |
| **restaurante\_categoria** | Entidad que guarda las categorías(tipo de comida) que sirve un restaurante. | **PK** | id | int |  | Id de restaurante\_categoria |
| FK | idRestaurante | int |  | Id del restaurante |
| FK | idCategoria | int |  | Id de la categoría |
|  | estado | int | 1 | Indica si restaurante\_categoria está activo o inactivo |
| **preferencia** | Entidad que almacena las categorías de preferencia del usuario. | **PK** | id | int |  | Id de la preferencia |
| FK | idUsuario | int |  | Id del usuario |
| FK | idCategoria | int |  | Id de la categoría |
|  | estado | int | 1 | Indica si la preferencia está activa o inactiva |

1. **DEFINICIÓN DE SERVICIOS**

Para nuestra aplicación hemos definido los siguientes servicios:

|  |  |
| --- | --- |
| Titulo | Obtener Restaurantes, lista todos u obtiene un restaurante. |
| URL | api/restaurantes |
| Método | GET |
| Parámetros del URL | Opcionales: idRestaurante (para devolver un restaurante específico) |
| Parámetros de datos |  |
| Respuesta Exitosa | Código: 200 (HTTP Response)  Contenido: { data:{ id: [int], nombre: [string], latitude: [decimal(11,8)], longitude: [decimal(11,8)], descripcion: [string], puntuacionTotal: [decimal(10,2)] } }  Ejemplo: { data: { id: 2, nombre: “Rustica”, idDistrito: 2, latitud: -77.0829897, longitud: -11.4897042, descripcion: “Restaurante con 21 años de creación”, puntuacionTotal: 8.3, estado: 1 } } |
| Respuesta de Error | Código: 401 UNAUTHORIZED  Contenido: {error: “Obtener Restaurantes”} |
| Ejemplo de Invocación | $.ajax({ url: " api/restaurantes", dataType: "json", type : "GET", success : function(r) { console.log(r); }}); |
| Notas |  |

|  |  |
| --- | --- |
| Titulo | Crear Restaurante, crea un restaurante. |
| URL | api/restaurantes/create |
| Método | POST |
| Parámetros del URL |  |
| Parámetros de datos | { nombre: [string], latitud : [decimal(11,8)] , longitud : [decimal(11,8)], descripcion: [string], puntuacionTotal: [decimal(10,2)] }}  Ejemplo: { data : { nombre: “La Ultima Cena”, latitud: -77.0829897, longitud: -11.4897042, descripcion: “restaurant bien bueno”, puntuacionTotal: 5.3 } |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { data : { id: [int], nombre: [string], latitud : [decimal(11,8)] , longitud : [decimal(11,8)], descripcion: [string], puntuacionTotal: [decimal(10,2)] }}  Ejemplo: { data : { id: 16, nombre: “La Ultima Cena”, latitud: -77.0829897, longitud: -11.4897042, descripcion: “restaurant bien bueno”, puntuacionTotal: 5.3 }} |
| Respuesta de Error | Código: 401 UNAUTHORIZED  Contenido: {error: “Crear Restaurante} |
| Ejemplo de Invocación | $.ajax({ url: " api/restaurantes/create", dataType: "json", type : "POST", success : function(r) { console.log(r); }}); |
| Notas |  |

|  |  |
| --- | --- |
| Titulo | Restaurantes por Categoría, obtiene todos los restaurantes de una categoría. |
| URL | api/categorías/{id}/restaurantes |
| Método | GET |
| Parámetros del URL |  |
| Parámetros de datos |  |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { data : { id: [int], nombre: [string], latitud : [decimal(11,8)] , longitud : [decimal(11,8)], descripcion: [string], puntuacionTotal: [decimal(10,2)], estado: [int] } … }  Ejemplo: { data : { id: 16, nombre: “La Ultima Cena”, latitud: -77.0829897, longitud: -11.4897042, descripcion: “restaurant bien bueno”, puntuacionTotal: 5.3, estado: 1 } … } |
| Respuesta de Error | Código: 401 UNAUTHORIZED  Contenido: {error: “Obtener Restaurantes por Categoria” } |
| Ejemplo de Invocación | $.ajax({ url: "api/categorías/8/restaurantes ", dataType: "json", type : "GET", success : function(r) { console.log(r); }}); |
| Notas |  |

|  |  |
| --- | --- |
| Titulo | Crear Recomendación, crea una nueva recomendación para un restaurante. |
| URL | api/recomendaciones/create |
| Método | POST |
| Parámetros del URL |  |
| Parámetros de datos | { comentario: [string], puntuación: [int], idUsuario : [int], idRestaurante: [int] , idGrupo: [int] }  Ejemplo: { comentario: “Este restaurante es casi perfecto!”, puntuación: 9, idUsuario : 26, idRestaurante: 14 , idReceptor: 9 } |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { data : { id: [int], comentario: [string], puntuación: [int] idUsuario : [int], idRestaurante: [int] , idGrupo: [int] } }  Ejemplo: { data : { id: 32, comentario: “Este restaurante es casi perfecto!”, puntuación: 9, idUsuario : 26, idRestaurante: 14 , idReceptor: 9 } } |
| Respuesta de Error | Código: 401 UNAUTHORIZED  Contenido: {error: “Crear Recomendacion” } |
| Ejemplo de Invocación | $.ajax({ url: " api/recomendaciones/create", dataType: "json", type : "POST", success : function(r) { console.log(r); }}); |
| Notas |  |

|  |  |
| --- | --- |
| Titulo | Obtener Recomendación, obtiene las recomendaciones hechas a un usuario. |
| URL | api/recomendaciones |
| Método | GET |
| Parámetros del URL | Opcional: idUsuario (para devolver una lista de recomendaciones hechas a un usuario específico) |
| Parámetros de datos |  |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { data : { id: [int], comentario: [string], puntuación: [int] idUsuario : [int], idRestaurante: [int] , idGrupo: [int], estado: [int] } … }  Ejemplo: { data : { id: 32, comentario: “Este restaurante es casi perfecto!”, puntuación: 9, idUsuario : 26, idRestaurante: 14 , idReceptor: 9, estado: 1 } … } |
| Respuesta de Error | Código: 401 UNAUTHORIZED  Contenido: {error: “Obtener Recomendación” } |
| Ejemplo de Invocación | $.ajax({ url: " api/recomendaciones ", dataType: "json", type : "GET", success : function(r) { console.log(r); }}); |
| Notas |  |

|  |  |
| --- | --- |
| Titulo | Crear Usuario, crea un usuario nuevo. |
| URL | api/usuarios/create |
| Método | POST |
| Parámetros del URL |  |
| Parámetros de datos | { nombres: [string], apellidos: [string] username : [string], emai: [string] , facebook\_id: [int], password: [string], is\_admin: [int] }  Ejemplo: { nombres: “Juan”, apellidos: “Perez” username : “jperez”, emai: “jperez25@gmail.com” , facebook\_id: 23, password: “qwerty”, idDistrito: 12, is\_admin: 0 } |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { data : { id: [int], nombres: [string], apellidos: [string] username : [string], emai: [string] , facebook\_id: [int], password: [string], is\_admin: [int] } }  Ejemplo: { data : { id: 25, nombres: “Juan”, apellidos: “Perez” username : “jperez”, emai: “jperez25@gmail.com” , facebook\_id: 23, password: “qwerty”, idDistrito: 12, is\_admin: 0 } } |
| Respuesta de Error | Código: 401 UNAUTHORIZED  Contenido: {error: “Crear Usuario” } |
| Ejemplo de Invocación | $.ajax({ url: "api/usuarios/create ", dataType: "json", type : "GET", success : function(r) { console.log(r); }}); |
| Notas |  |

|  |  |
| --- | --- |
| Titulo | Actualizar Usuario, actualiza un usuario existente. |
| URL | api/usuarios/{id} |
| Método | PUT |
| Parámetros del URL |  |
| Parámetros de datos | { nombres: [string], apellidos: [string] username : [string], emai: [string] , facebook\_id: [int], password: [string], is\_admin: [int] }  Ejemplo: { nombres: “JuanEditado”, apellidos: “PerezEditado” username : “jperez”, emai: “jperez25@gmail.com” , facebook\_id: 23, password: “qwerty2”, idDistrito: 12, is\_admin: 0 } |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { data : { id: [int], nombres: [string], apellidos: [string] username : [string], emai: [string] , facebook\_id: [int], password: [string], is\_admin: [int] } }  Ejemplo: { data : { id: 25, nombres: “JuanEditado”, apellidos: “PerezEditado” username : “jperez”, emai: “jperez25@gmail.com” , facebook\_id: 23, password: “qwerty2”, idDistrito: 12, is\_admin: 0 } } |
| Respuesta de Error | Código: 401 UNAUTHORIZED  Contenido: {error: “Actualizar Usuario” } |
| Ejemplo de Invocación | $.ajax({ url: "api/usuarios/2 ", dataType: "json", type : "PUT", success : function(r) { console.log(r); }}); |
| Notas |  |

|  |  |
| --- | --- |
| Titulo | Eliminar Usuario, elimina un usuario existente. |
| URL | api/usuarios/{id} |
| Método | DELETE |
| Parámetros del URL |  |
| Parámetros de datos |  |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { data : { id: [int], message: [string]} }  Ejemplo: { data : { id: 25, message: “soft deleted” } } |
| Respuesta de Error | Código: 401 UNAUTHORIZED  Contenido: {error: “Eliminar Usuario” } |
| Ejemplo de Invocación | $.ajax({ url: "api/usuarios/3", dataType: "json", type : "DELETE", success : function(r) { console.log(r); }}); |
| Notas |  |

|  |  |
| --- | --- |
| Titulo | Obtener Usuarios, obtiene todos o un usuario específico. |
| URL | api/usuarios/ |
| Método | GET |
| Parámetros del URL | Opcional: idUsuario (para devolver un usuario específico) |
| Parámetros de datos |  |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { data : { id: [int], nombres: [string], apellidos: [string] username : [string], emai: [string] , facebook\_id: [int], password: [string], is\_admin: [int], estado: [int] } … }  Ejemplo: { data : { id: 25, nombres: “Juan”, apellidos: “Perez” username : “jperez”, emai: “jperez25@gmail.com” , facebook\_id: 23, password: “qwerty”, idDistrito: 12, is\_admin: 0, estado: 1 } … } |
| Respuesta de Error | Código: 401 UNAUTHORIZED  Contenido: {error: “Obtener Usuario” } |
| Ejemplo de Invocación | $.ajax({ url: "api/usuarios ", dataType: "json", type : "GET", success : function(r) { console.log(r); }}); |
| Notas |  |

|  |  |
| --- | --- |
| Titulo | Obtener Categorías, obtiene todas o una categoría específica. |
| URL | api/categorias |
| Método | GET |
| Parámetros del URL | Opcional: idCategoria (para devolver una categoría específica) |
| Parámetros de datos |  |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { data : { id: [int], nombre: [string], descripcion: [string], estado: [int] } … }  Ejemplo: { data: { id: 7, nombre: “criolla”, descripcion: “comida peruana”, estado: 1 } … } |
| Respuesta de Error | Código: 401 UNAUTHORIZED  Contenido: {error: “Obtener Categoria” } |
| Ejemplo de Invocación | $.ajax({ url: "api/categorias ", dataType: "json", type : "GET", success : function(r) { console.log(r); }}); |
| Notas |  |

|  |  |
| --- | --- |
| Titulo | Obtener Preferencias, obtiene todas las preferencias de un usuario. |
| URL | api/preferencias |
| Método | GET |
| Parámetros del URL | Obligatorio: idUser (para obtener las preferencias del usuario con ese código) |
| Parámetros de datos |  |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { data : { id: [int], idUsuario: [int], idCategoria: [int], estado: [int] } … }  Ejemplo: { data: { id: 7, idUsuario: 12, idCategoria: 8, estado: 1 } … } |
| Respuesta de Error | Código: 401 UNAUTHORIZED  Contenido: {error: “Obtener Preferencias” } |
| Ejemplo de Invocación | $.ajax({ url: "api/preferencias ", dataType: "json", type : "GET", success : function(r) { console.log(r); }}); |
| Notas |  |

|  |  |
| --- | --- |
| Titulo | Crear Preferencia, crea una preferencia para un usuario. |
| URL | api/preferencias/create |
| Método | POST |
| Parámetros del URL |  |
| Parámetros de datos | { idUsuario: [int], idCategoria: [int] }  Ejemplo: { idUsuario: 18, idCategoria: 6 } |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { data : { id: [int], idUsuario: [int], idCategoria: [int], estado: [int] } }  Ejemplo: { data: { id: 9, idUsuario: 14, idCategoria: 7, estado: 1 } } |
| Respuesta de Error | Código: 401 UNAUTHORIZED  Contenido: {error: “Crear Preferencia” } |
| Ejemplo de Invocación | $.ajax({ url: "api/preferencias/create ", dataType: "json", type : "POST", success : function(r) { console.log(r); }}); |
| Notas |  |

|  |  |
| --- | --- |
| Titulo | Eliminar Preferencia, elimina una preferencia existente para un usuario. |
| URL | api/preferencias |
| Método | DELETE |
| Parámetros del URL |  |
| Parámetros de datos |  |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { data : { id: [int], message: [string]} }  Ejemplo: { data : { id: 25, message: “soft deleted” } } |
| Respuesta de Error | Código: 401 UNAUTHORIZED  Contenido: {error: “Eliminar Preferencia” } |
| Ejemplo de Invocación | $.ajax({ url: "api/preferencias ", dataType: "json", type : "DELETE", success : function(r) { console.log(r); }}); |
| Notas |  |

|  |  |
| --- | --- |
| Titulo | Login, verifica las credenciales del usuario. |
| URL | api/usuario/login |
| Método | POST |
| Parámetros del URL |  |
| Parámetros de datos | { username: [string], password: [string] }  Ejemplo: { username: juanito123, password: contra123 } |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { data : { id: [int], nombres: [string], apellidos: [string] username : [string], emai: [string] , facebook\_id: [int], password: [string], is\_admin: [int] } }  Ejemplo: { data : { id: 25, nombres: “JuanEditado”, apellidos: “PerezEditado” username : “jperez”, emai: “jperez25@gmail.com” , facebook\_id: 23, password: “qwerty2”, idDistrito: 12, is\_admin: 0 } } |
| Respuesta de Error | Código: 401 UNAUTHORIZED  Contenido: {error: “Usuario no identificado” } |
| Ejemplo de Invocación | $.ajax({ url: "api/usuario/login ", dataType: "json", type : "POST", success : function(r) { console.log(r); }}); |
| Notas |  |

|  |  |
| --- | --- |
| Titulo | Listar Grupos, lista los grupos existentes. |
| URL | api/usuarios/{id}/grupos |
| Método | GET |
| Parámetros del URL |  |
| Parámetros de datos |  |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { data : { id: [int], nombre: [string], created\_by: [string] , updated\_by: [int],  estado: [int] } }  Ejemplo: { data : { id: 1, nombre: “amigos del trabajo”, created\_by: 1 , updated\_by: 1,  estado: 1 } … } |
| Respuesta de Error |  |
| Ejemplo de Invocación | $.ajax({ url: "api/usuarios/grupos ", dataType: "json", type : "GET", success : function(r) { console.log(r); }}); |
| Notas |  |

|  |  |
| --- | --- |
| Titulo | Crear Grupos, crea grupo para un usuario. |
| URL | api/grupos/create |
| Método | POST |
| Parámetros del URL |  |
| Parámetros de datos | { nombre: [string], created\_by: [int] }  Ejemplo: { nombre: “Familia”, created\_by: 1 } |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { "data": {  "id": [int],  "nombre": [string],  "created\_by": [int],  "updated\_by": [int],  “estado”: [int]  } }  Ejemplo: { "data": {  "id": 3,  "nombre": "Mi nuevo grupo",  "created\_by": 1,  "updated\_by": 1,  “estado”:1  } } |
| Respuesta de Error |  |
| Ejemplo de Invocación | $.ajax({ url: "api/grupos/create ", dataType: "json", type : "POST", success : function(r) { console.log(r); }}); |
| Notas |  |
| Titulo | Crear foto**, sube una foto al servidor de Flickr para que sea usada por la aplicación.** |
| URL | api/fotos/create |
| Método | POST |
| Parámetros del URL |  |
| Parámetros de datos | { nombre: [string], foto: [file] }  Ejemplo: { nombre: myRestaurantPhoto, file: imagen.jpg } |
| Respuesta Exitosa | Código: 200 (HTTP Response)  { data : { id: [int], nombre: [string], url: [string], formato: [string], estado: [int] } }  Ejemplo: { data : { id: 3, nombre: myRestaurantPhoto, url: , formato: JPG, estado: 1 } } |
| Respuesta de Error | Código: 401 UNAUTHORIZED  Contenido: {error: “Crear foto” } |
| Ejemplo de Invocación | $.ajax({ url: "api/ fotos/create", dataType: "json", type : "POST", success : function(r) { console.log(r); }}); |
| Notas |  |

**Facebook SDK for Android**

El SDK de Facebook para Android es la manera más fácil de integrar nuestra aplicación RestaurApp con Facebook. Este SDK permitirá a nuestra aplicación:

* **Facebook Login**

Permite la autenticación con credenciales de Facebook. Permitirá a la aplicación usar el siguiente servicio:

|  |  |
| --- | --- |
| Titulo | Facebook Login, verifica las credenciales del usuario utilizando su cuenta de facebook. |
| URL |  |
| Método |  |
| Parámetros del URL |  |
| Parámetros de datos | Permisos de la aplicación [String]  loginButton=(LoginButton)view.findViewById(R.id.login\_button);  loginButton.setReadPermissions("user\_friends"); |
| Respuesta Exitosa | Se ejecutará la función onSuccess(LoginResult loginResult) |
| Respuesta de Error | Se ejecutará la función onError(FacebookException expception) |
| Ejemplo de Invocación | Una vez configurado el SDK, se llamará al login de Facebook mediante un botón:  <com.facebook.login.widget.LoginButton  android:id="@+id/login\_button"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_gravity="center\_horizontal"  android:layout\_marginTop="30dp"  android:layout\_marginBottom="30dp" /> |
| Notas | Revisar la documentación oficial del login con Facebook utilizando Facebook SDK for Android: <https://developers.facebook.com/docs/facebook-login/android/v2.3> |

* **Compartir y enviar diálogos**

Las personas pueden compartir contenido de la aplicación a Facebook. Permitirá a la aplicación usar el siguiente servicio:

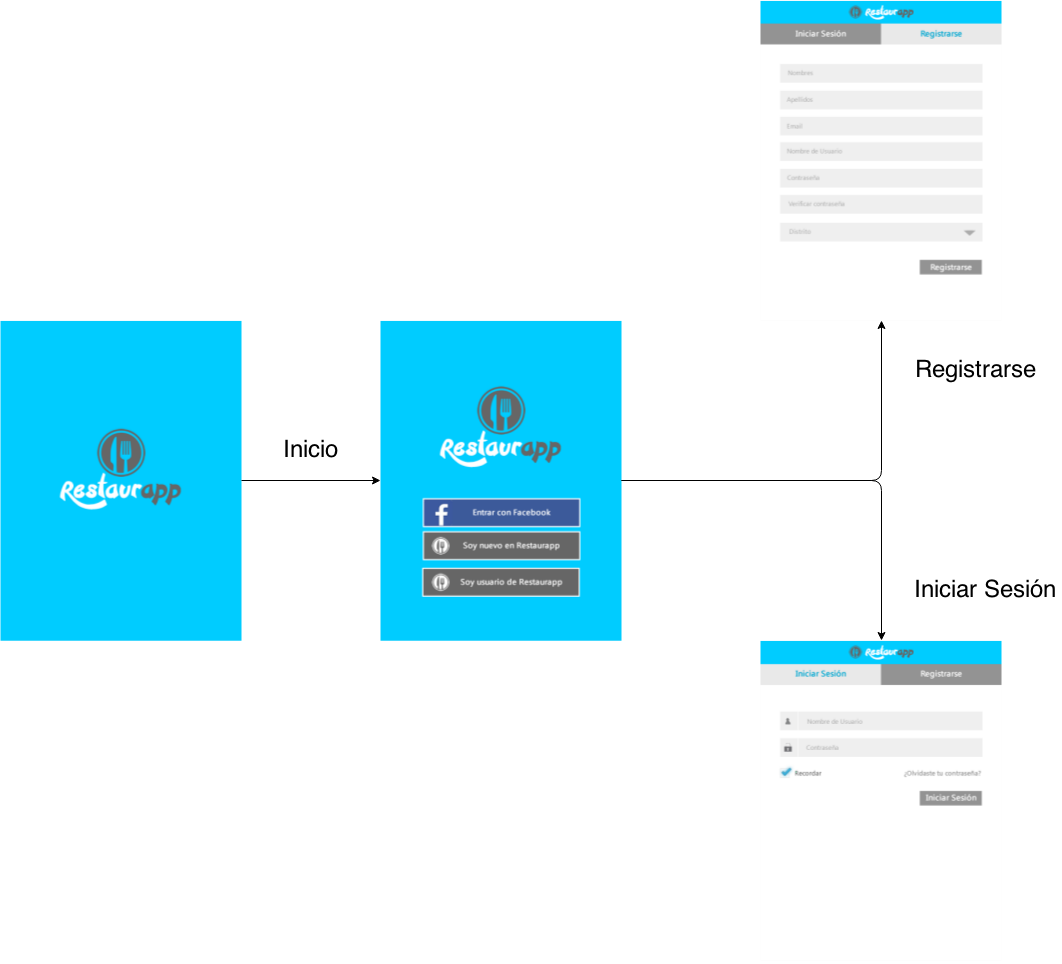
|  |  |
| --- | --- |
| Titulo | Recomendar Restaurante, permite al usuario compartir en Facebook un restaurante. |
| URL |  |
| Método |  |
| Parámetros del URL |  |
| Parámetros de datos | * contentURL, el link a ser compartido * contentTitle, representa el título del contenido del link * imageURL, la URL o imagen miniatura que aparecerá en el post * contentDescription, 2 a 4 oraciones que describan el contenido   Ejemplo:  ShareLinkContent content = new ShareLinkContent.Builder() .setContentUrl(Uri.parse("https://developers.facebook.com")).build(); |
| Respuesta Exitosa |  |
| Respuesta de Error |  |
| Ejemplo de Invocación | ShareButton shareButton = (ShareButton)findViewById(R.id.fb\_share\_button);  shareButton.setShareContent(content); |
| Notas | Revisar la documentación oficial de compartir con Facebook utilizando Facebook SDK for Android: <https://developers.facebook.com/docs/sharing/android>  Hay varias formas de compartir, en la documentación están los ejemplos. |

Reference: <https://developers.facebook.com/docs/android>

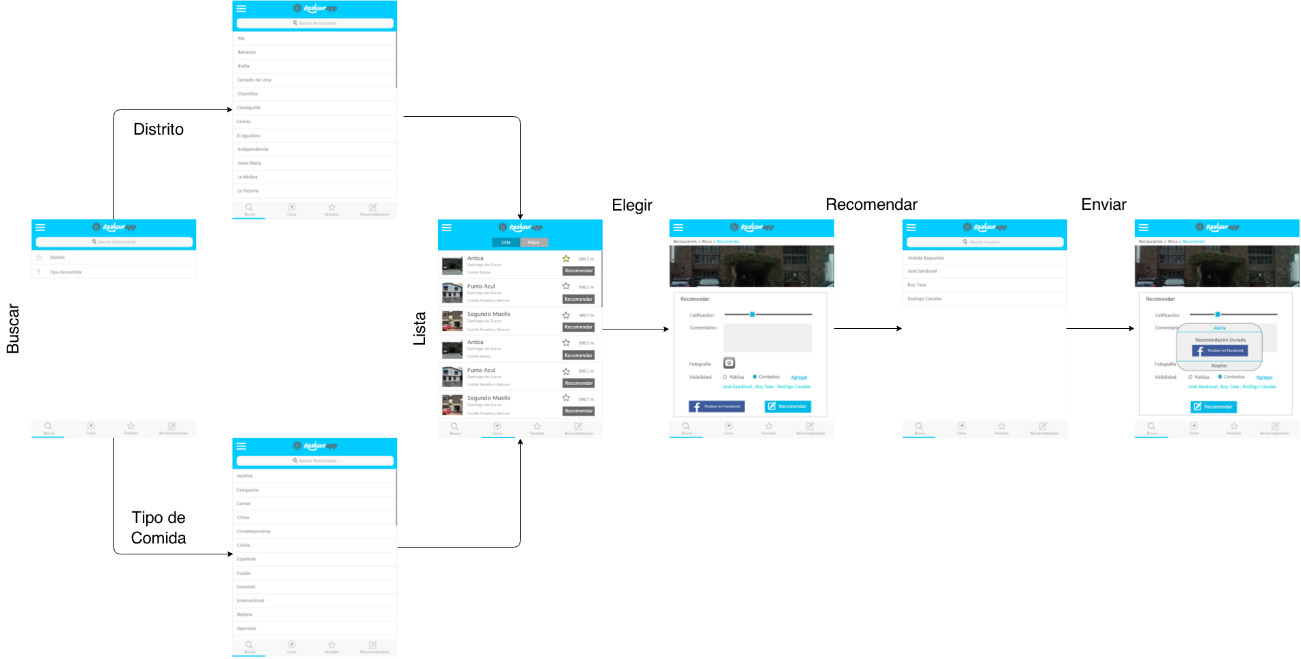
1. **FLUJOGRAMA DE LA APLICACIÓN MÓVIL**

A continuación se presenta el flujograma de la aplicación móvil:

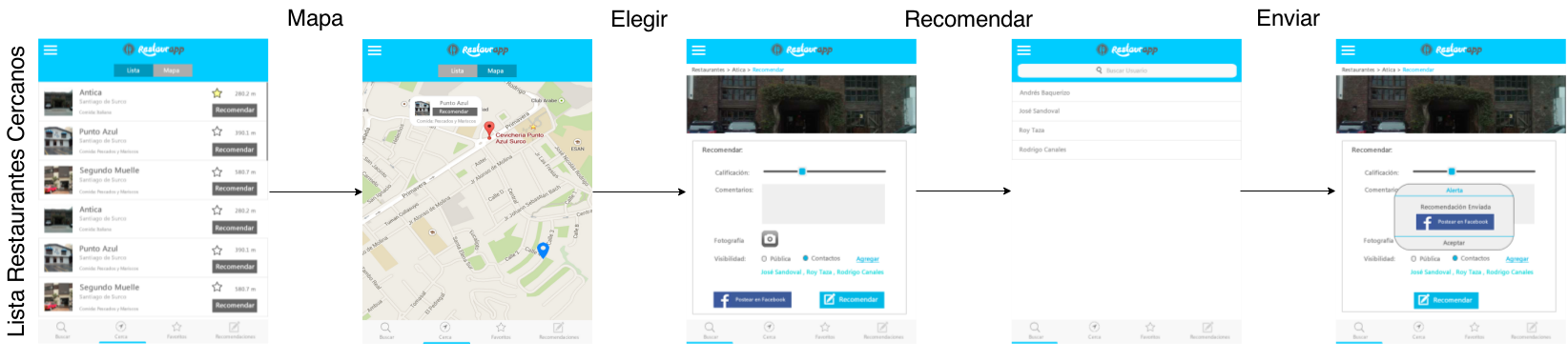
En la siguiente imagen se pueden apreciar las pantallas de Inicio, Registro e Inicio de sesión.



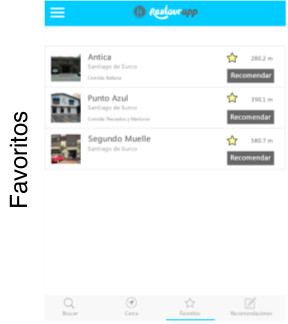
Luego de ingresar con nuestro usuario, se nos presentarán 4 opciones principales: “Buscar”, “Cerca”, “Favoritos” y “Recomendaciones”. Estas opciones pueden ser seleccionadas en la parte inferior. La opción predeterminada que se muestra es “Buscar”, se pueden buscar restaurantes por “Distrito” y “Tipo de Comida”. Se nos mostrará una lista de la cual podremos elegir un restaurante para recomendar a un amigo incluyendo fotos, comentarios y ubicación.



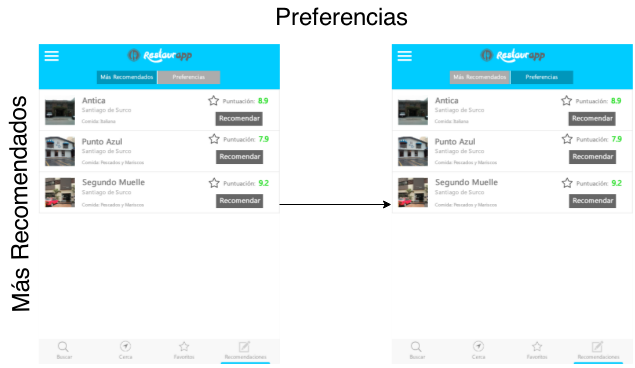
La opción “Cerca” nos muestra los restaurantes más cercanos a nuestra ubicación. Tenemos la opción “Mapa” que permite visualizar la ubicación del restaurante con Google Maps. Aquí también podremos elegir el restaurante y hacer una recomendación.



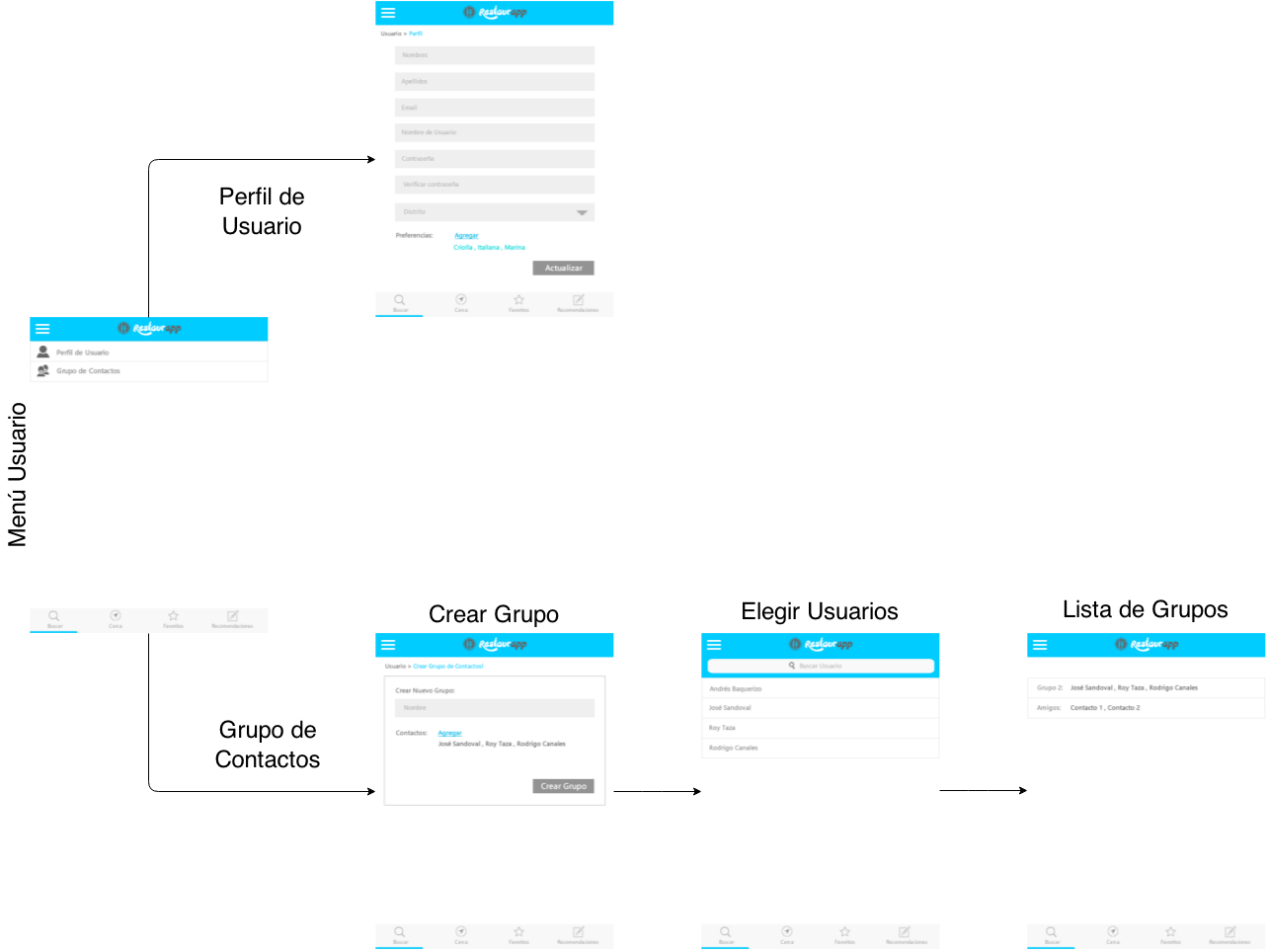
La tercera opción es “Favoritos”, una pantalla que nos muestra los restaurantes que hayamos marcado como favoritos anteriormente.



La opción “Recomendaciones” nos muestra las recomendaciones que otros usuarios nos han enviado, con el botón “Preferencias” podremos obtener una lista con los restaurantes mejor puntuados que tengan relación con las preferencias elegidas en nuestro perfil.



Finalmente tenemos el menú de usuario. En esta pantalla podremos acceder a nuestro perfil de usuario y actualizar nuestros datos personales así como nuestras preferencias por tipo de comida. También podemos crear un grupo de contactos para posteriormente poder mandar una recomendación a varias personas sin tener que hacer una recomendación por cada uno.



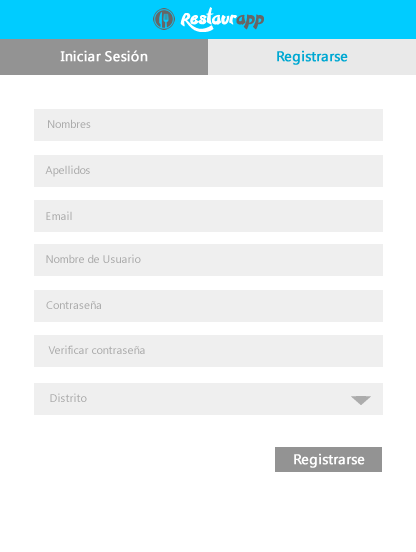
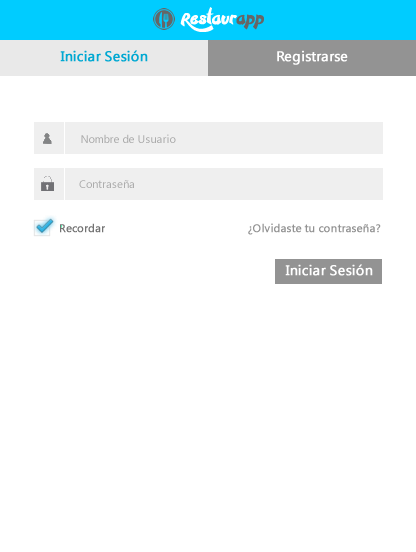
1. **MOCKUPS DE APLICACIÓN MÓVIL**

A continuación, se presentarán los prototipos de la aplicación móvil RestaurApp. Para una mejor comprensión de las interfaces, por favor revisar el flujograma de la aplicación web (sección 7).

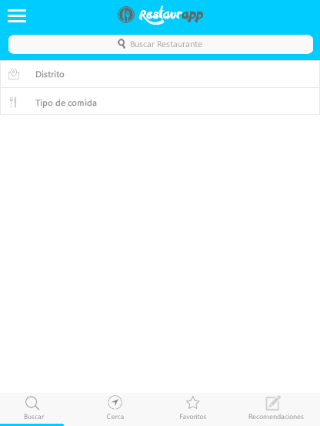
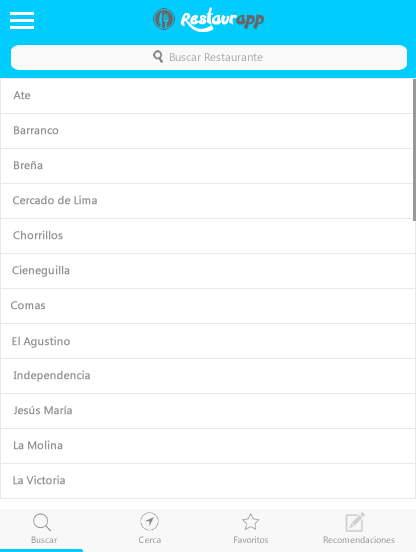
Pantalla de bloqueo Pantalla de Inicio

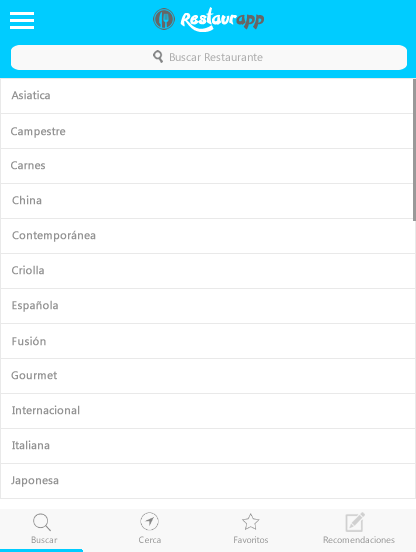
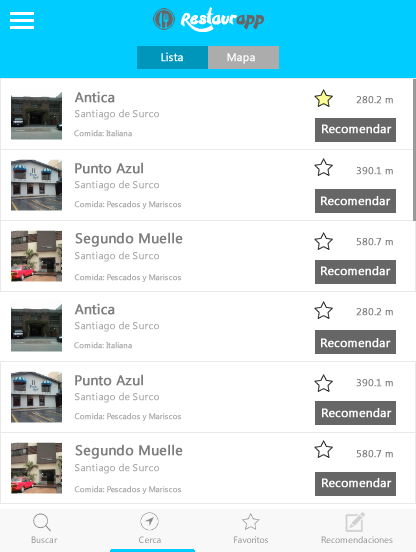
Registro de usuario Login usuario

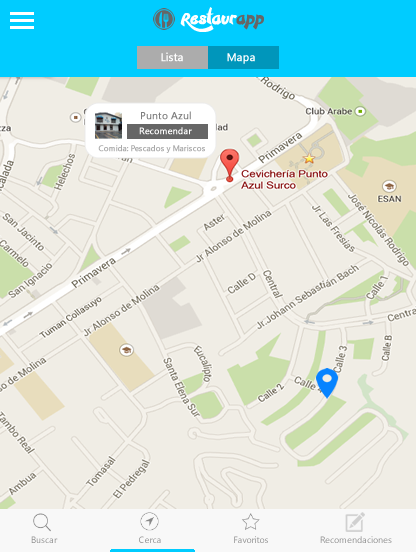
Búsqueda de restaurantes Búsqueda por distrito

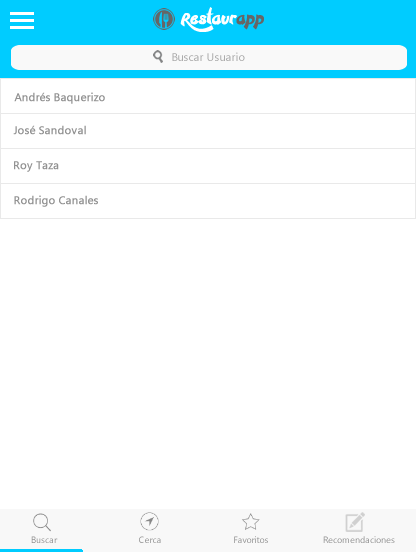
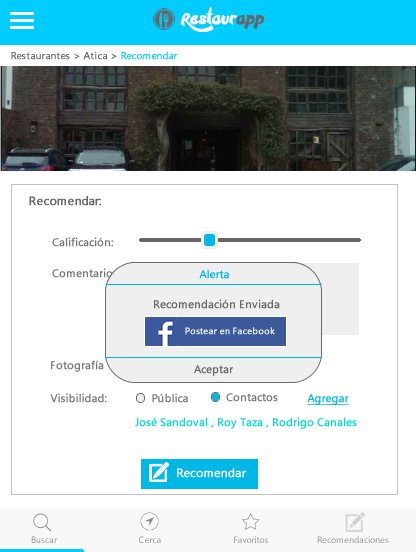
Búsqueda por categoría Restaurantes cercanos

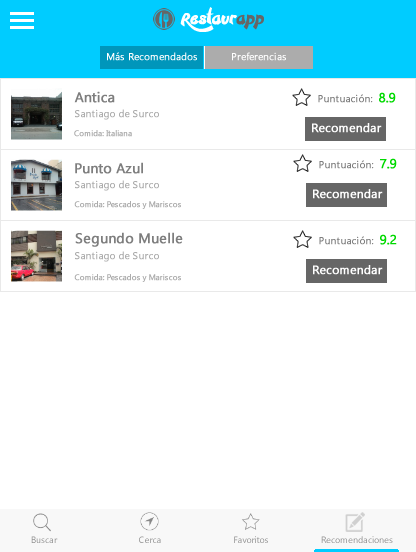
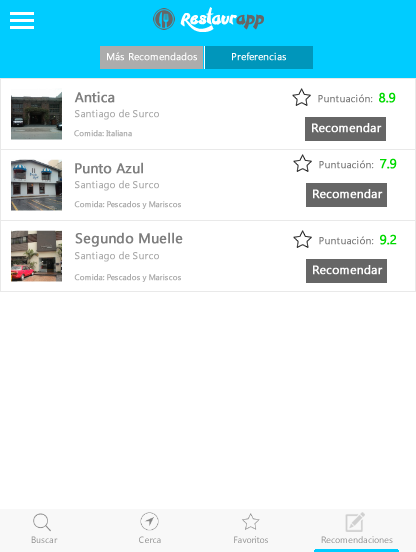
Restaurante en mapa Recomendar un restaurante

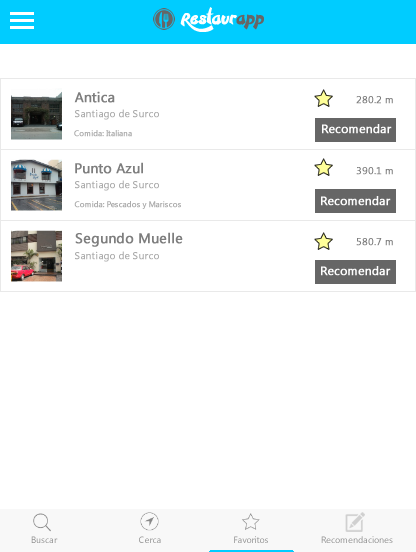
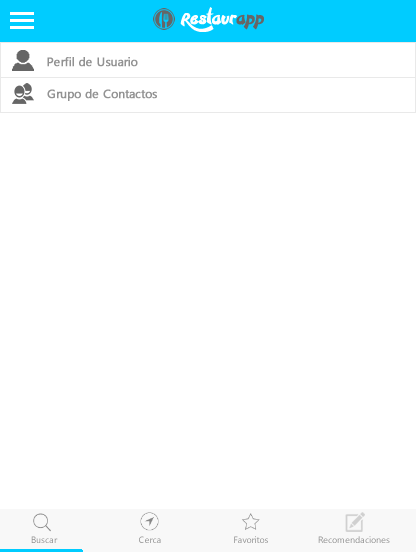
Buscar usuario Compartir en Facebook

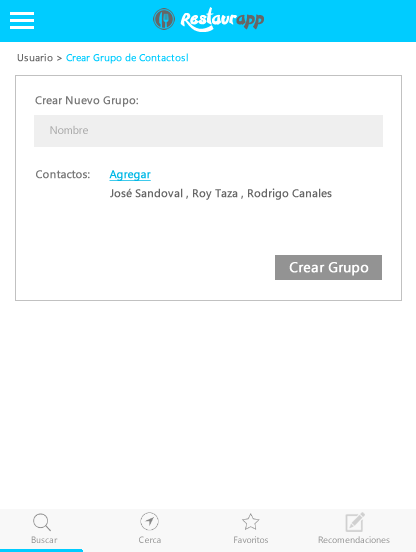
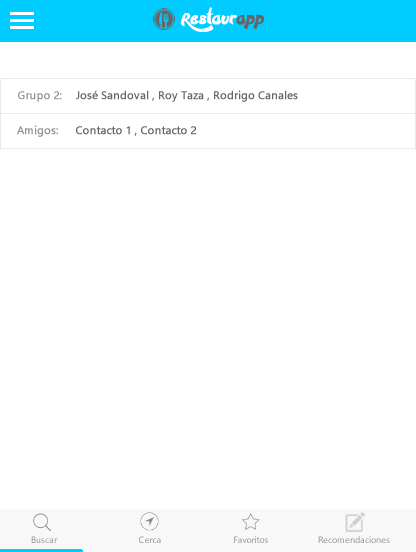
Restaurantes recomendados Restaurantes sugeridos

Restaurante favorito Búsqueda de contactos

Crear grupo de contacto Grupo de amigos

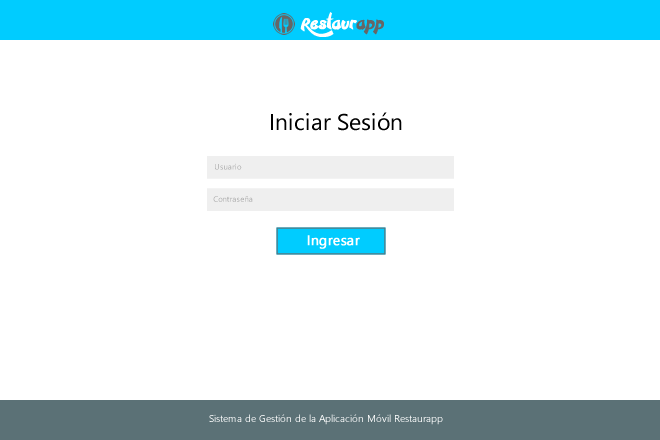
 

1. **MOCKUPS DE APLICACIÓN WEB**

A continuación, se presentarán los mockups de la aplicación web de RestaurApp, la cual estará orientado a la administración del sistema.

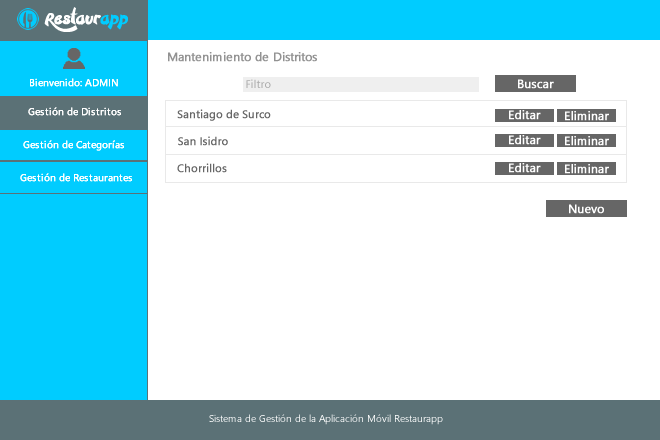
**Iniciar Sesión**

Pantalla para el logueo del administrador.

****

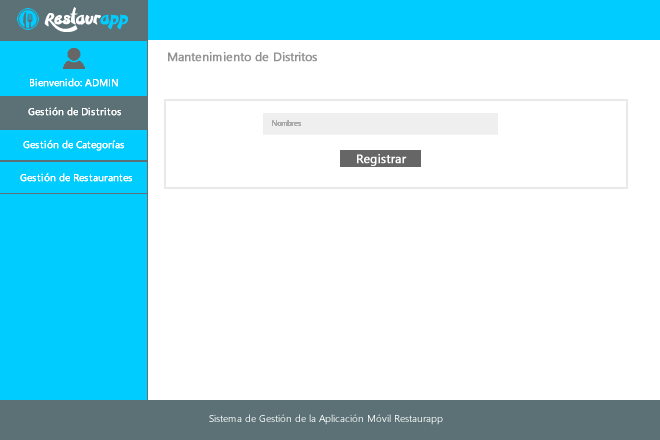
**Mantenimiento de distritos**

En esta interfaz el administrador realizará el mantenimiento de distritos.



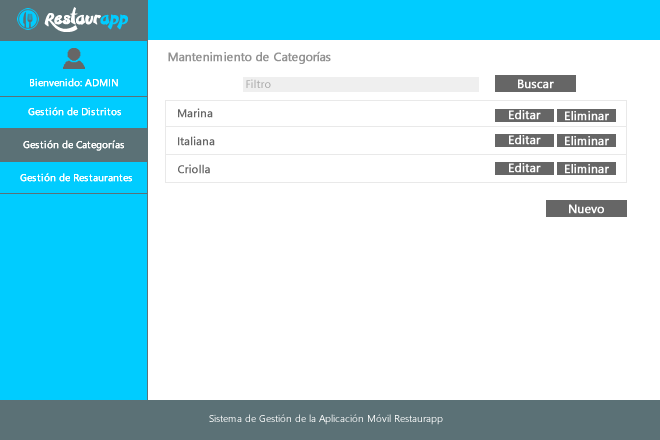
**Registrar distrito** y **Editar distrito**

Interfaz para registrar un distrito. Asimismo esta interfaz se utilizará para editar distrito.



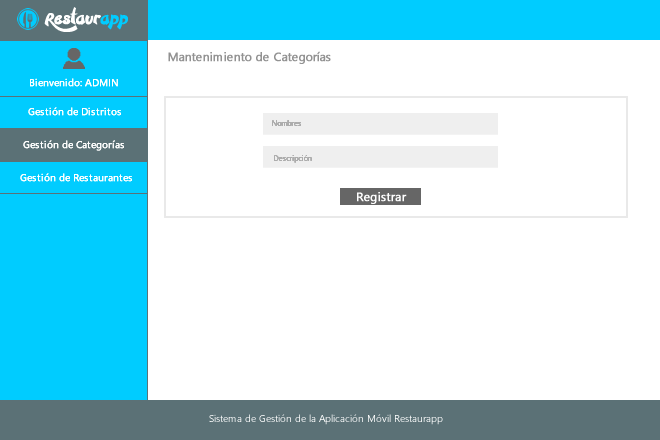
**Mantenimiento de categorías**

Interfaz en la cual el administrador puede listar, buscar, editar y eliminar una categoría.



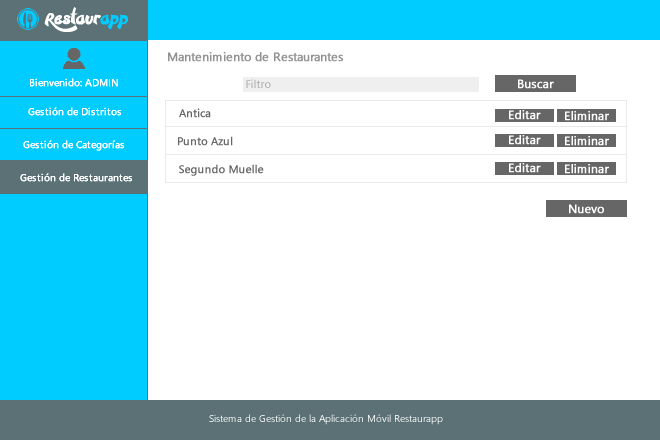
**Registrar categoría** y **Editar categoría**

Interfaz para registrar una nueva categoría. Asimismo, se utilizará esta interfaz para editar una categoría ya existente.



**Mantenimiento de restaurantes**

Pantalla para la administración de restaurantes, con las opciones de buscar, editar, eliminar y nuevo.



**Registrar restaurante** y **Editar restaurante**

Interfaz para registrar un nuevo restaurante. Asimismo, se utilizará esta interfaz para editar un restaurante ya existente.

