

Instituto Tecnológico de Costa Rica

Escuela de ingeniería en computación

Desarrollo de aplicaciones para dispositivos móviles

IC-8066

Proyecto 1:

Restaurantes

Bryan Brenes Rojas - 201173608

Jean Pierre Monterrey Arguedas - 2016246285

Prof.: Andrei Fuentes Leiva

Semestre I

24 de abril de 2019

Índice

1. Introducción	3
2. Wireframes	3
2.1. Pantalla inicial	3
2.2. Pantalla de registro	3
2.3. Pantalla de login	4
2.4. Pantalla de restablecimiento de contraseña	5
2.5. Pantalla principal	5
2.6. Pantalla para detalles del restaurante	5
3. Diseño de alto nivel de la aplicación	7
4. Web service	8
4.1. Usuarios	8
4.1.1. Registro de usuario: /createUser	8
4.1.2. Registro de usuario: /loginUser	8
4.1.3. Registro de usuario: /requestCode	9
4.1.4. Registro de usuario: /resetPassword	9
4.1.5. Registro de usuario: /updateName	10
4.1.6. Registro de usuario: /updatePhoto	10
4.2. Comidas	11
4.2.1. Registro de usuario: /getFoods	11
4.3. Restaurantes	11
4.3.1. Registro de usuario: /createRestaurant	11
4.3.2. Registro de usuario: /updateRestaurant	12
4.3.3. Registro de usuario: /createOpinion	12
4.3.4. Registro de usuario: /getOpinions	13
4.3.5. Registro de usuario: /addPhoto	13
4.3.6. Registro de usuario: /getPhotos	14
4.3.7. Registro de usuario: /getRestaurants	14

1. Introducción

Se desarrolla una aplicación en android para familiarizarse con el desarrollo en esta plataforma. A grandes rasgos, la aplicación debe permitir al usuario ingresar nuevos restaurantes donde se especifique información relevante entre esta la dirección que deberá ser mostrada en un mapa utilizando el API de Google para dicho fin. Además se debe poder listar y mostrar detalles de los restaurantes, así como poder realizar y visualizar comentarios del restaurante.

Se implementa además un backoffice para el manejo de las funciones de los administradores, así como un API para manejar el backend de la aplicación, lo que incluye el manejo de una base de datos en MongoDB y el manejo de consultas mediante el uso de json tanto para la solicitud como para la respuesta.

2. Wireframes

Con el objetivo de realizar la interfaz gráfica de la aplicación de una manera más precisa y eficiente se desarrollaron *wireframes* donde se especifica la distribución de los elementos gráficos así como una idea de las proporciones que estos tendrán en cada una de las pantallas de la aplicación.

2.1. Pantalla inicial

Esta es la primer pantalla que ve el usuario. Se muestran dos opciones posibles: registrarse o ingresar a la aplicación. Además se incluye una imagen que va a contener el logo principal de la aplicación. Cada una de las opciones tiene un botón asociado del cual registrarse se considera como el botón principal, dejando ingresar como botón secundario. En la fig. 1 se muestra el *wireframe* para esta pantalla.

2.2. Pantalla de registro

Esta pantalla (fig. 2) mantiene la misma imagen que la pantalla inicial de manera que dé uniformidad a la aplicación. La pantalla contiene la información necesaria para que el usuario pueda ser registrado en el sistema, esta información incluye el nombre de usuario, el correo electrónico, la contraseña y la confirmación de la contraseña. Por último se tiene el botón de registro.

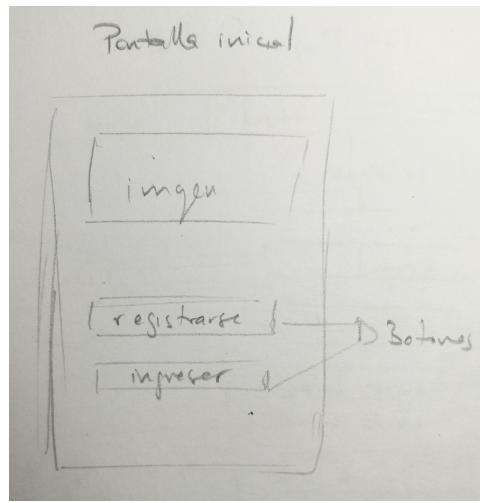


Figura 1: Pantalla inicial. Fuente: propia

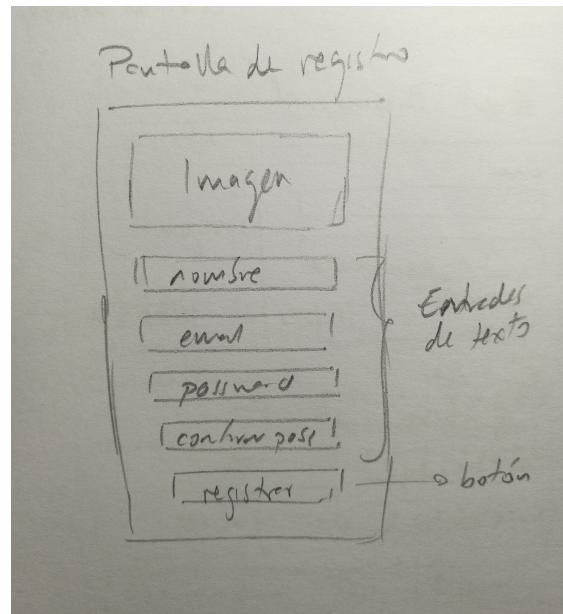


Figura 2: Pantalla de registro de usuario. Fuente: propia

2.3. Pantalla de login

Para realizar el login se requiere del correo electrónico del usuario así como su contraseña, por esto en el *wireframe* de esta pantalla que se encuentra en la fig. 3 se muestran los campos para ingresar el correo y la contraseña. Además se incluyen la opción para realizar el log in y para restablecer la contraseña en caso de olvido.

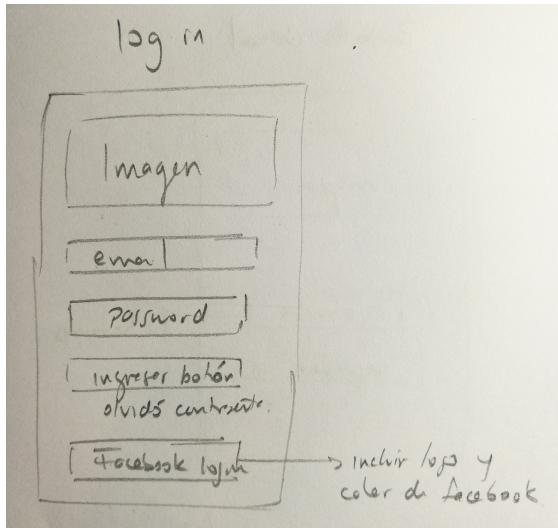


Figura 3: Pantalla de login de usuario. Fuente: propia

2.4. Pantalla de restablecimiento de contraseña

En la fig. 4 se muestran los elementos gráficos requeridos para realizar este proceso. Se coloca un campo de texto para que el usuario ingrese el correo electrónico al cual se le va a enviar el correo así como un botón para realizar el envío. Se dispone además de campos para la nueva contraseña así como para la confirmación de esta. Por último se tiene campo para ingresar el código que es enviado por el correo.

2.5. Pantalla principal

La pantalla principal de la aplicación está diseñada utilizando pestañas de manera que se tiene dos vista como se muestra en la fig. 5. Una de las pestañas tiene la vista del mapa con las ubicaciones de los restaurantes, mientras que la otra pestaña tiene una lista de todos los restaurantes, en esta lista se maneja un formato para cada uno de sus ítems, como se muestra en el *wireframe* se tiene una imagen del restaurante, nombre y ubicación, así como una imagen para mostrar la calificación del restaurante.

2.6. Pantalla para detalles del restaurante

Esta pantalla muestra la información correspondiente al restaurante seleccionado. Primero se muestra el nombre del restaurante seguido por las fotos que los usuarios han ingresado. Poste-

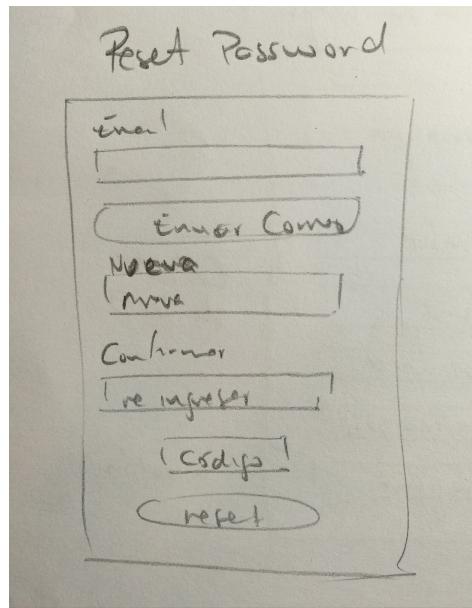


Figura 4: Pantalla de restablecimiento de contraseña. Fuente: propia

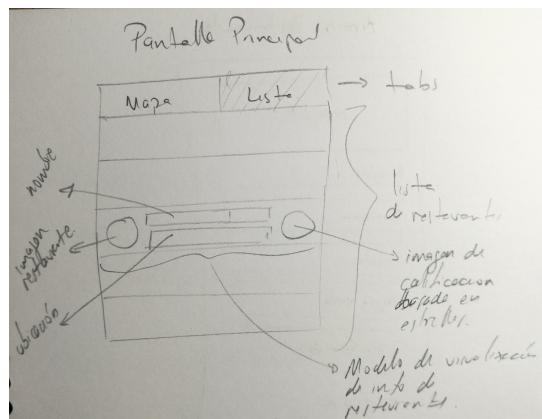


Figura 5: Pantalla principal. Fuente: propia

riormente mediante el uso de estrella se indica la puntuación promedio de este, además se indica mediante algún ícono el precio general de la comida. Seguido se muestra la información como teléfono, tipo de comida que venden y horarios. Luego se muestra en un mapa la ubicación del restaurante. En esta pantalla se permite al usuario ingresar un comentario, por lo que se incluye un conjunto de estrellas para que el usuario pueda dar su calificación, de igual manera se maneja la calificación del precio, por último se tiene un cuadro de texto para indicar el comentario. Como último elemento de la pantalla, se tiene una lista de los comentarios que los usuarios han realizado de dicho restaurante.

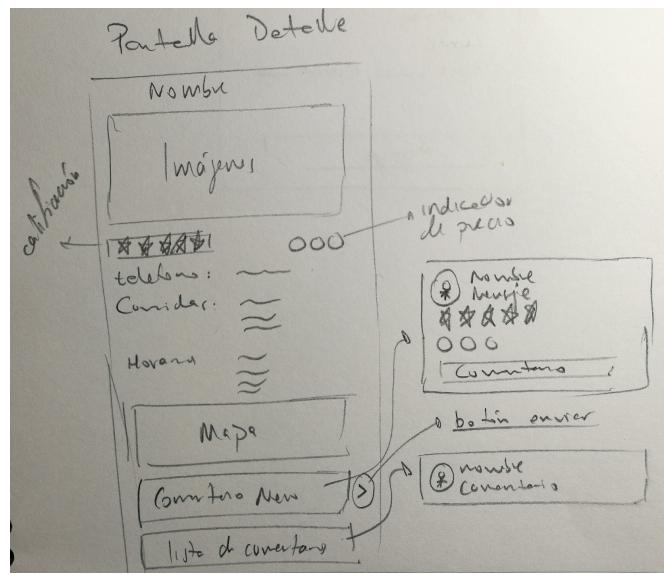


Figura 6: Pantalla para detalles del restaurante. Fuente: propia

3. Diseño de alto nivel de la aplicación

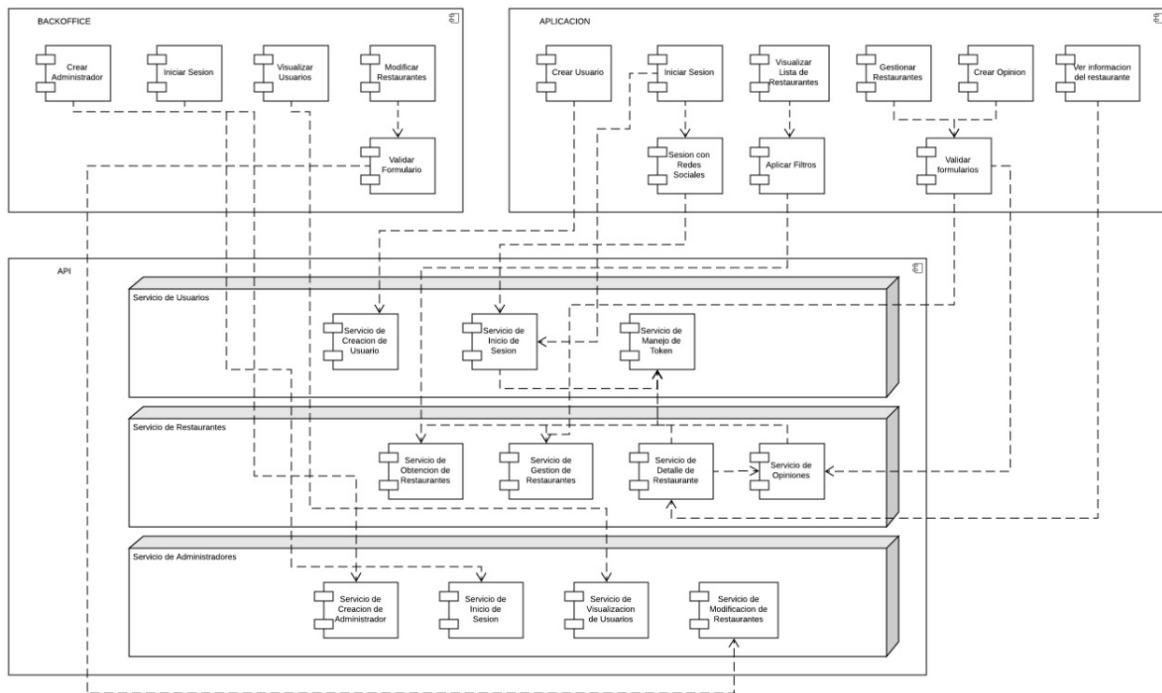


Figura 7: Diagrama para el desarrollo de la aplicación. Fuente: propia

4. Web service

La implementación del API para la aplicación fue desarrollada utilizando Node.js mediante la creación de las rutas necesarias para suplir las funcionalidad de la aplicación, entre las que están la creación, modificación y consulta de restaurantes, el manejo de usuarios con respecto a la creación y login, así como el manejo de comentarios, entre otras cosas.

Este API fue desplegado utilizando la plataforma Heroku por su sencillez de implementación, así como por ser gratuita. La ruta utilizada por el API es <https://foodcourtec.herokuapp.com/>

Las siguiente secciones muestran los módulos en los que se dividieron las funcionalidades y rutas utilizadas por el API.

4.1. Usuarios

4.1.1. Registro de usuario: /createUser

Envío:

```
{  
  "email": "Email del usuario"  
  "name": "Nombre del usuario"  
  "password": "Contraseña del usuario"  
}
```

Respuesta:

```
{  
  "status": "Mensaje según lo sucedido"  
}
```

4.1.2. Registro de usuario: /loginUser

Envío:

```
{  
  "email": "Email del usuario"  
  "name": "Nombre del usuario"  
  "SocialLogin": Boolean si el usuario ingresa con Facebook  
}
```

En este caso, si se inicia sesión con Facebook, la key “password” no es necesaria, igual en caso contrario.

Respuesta:

```
{  
  "token": "Token para las próximas solicitudes"  
  "name": "Nombre del usuario"  
  "photo": "Información binaria de la foto"  
  "contentType": "Formato de la foto"  
}
```

Si el inicio de sesión es por Facebook se recibirán únicamente el Token como respuesta.

4.1.3. Registro de usuario: /requestCode

Envío:

```
{  
  "email": "Email del usuario"  
}
```

Respuesta:

```
{  
  "status": "Mensaje según lo sucedido"  
}
```

Recordar que se envía un correo al email brindado con el código para cambiar la contraseña.

4.1.4. Registro de usuario: /resetPassword

Envío:

```
{  
  "email": "Email del usuario"  
  "password": "Contraseña nueva"  
  "code": "Código enviado al correo"  
}
```

Respuesta:

```
{
```

```
        "status": "Mensaje según lo sucedido"  
    }  
}
```

4.1.5. Registro de usuario: /updateName

Envío:

```
{  
    "email": "Email del usuario"  
    "token": "Token para solicitudes"  
    "name": "Nuevo nombre"  
}
```

Respuesta:

```
{  
    "token": "Token de respuesta"  
    "status": "Mensaje según lo sucedido"  
}
```

4.1.6. Registro de usuario: /updatePhoto

Envío:

```
{  
    "token": "Token para solicitudes"  
    "email": "Email del usuario"  
    "contentType": "Formato de la imagen"  
    "avatar": "Imagen a colocar"  
}
```

Respuesta:

```
{  
    "token": "Token de respuesta"  
    "status": "Mensaje según lo sucedido"  
}
```

4.2. Comidas

4.2.1. Registro de usuario: /getFoods

Envío:

```
{  
  "token": "Token para solicitudes"  
  "email": "Email del usuario"  
}
```

Respuesta:

```
{  
  "token": "Token de respuesta"  
  "foods": Comidas  
}
```

4.3. Restaurantes

Para los horarios, el array debe tener un tamaño exacto de 7 y cada dia debe tener el formato indicado.

4.3.1. Registro de usuario: /createRestaurant

Envío:

```
{  
  "token": "Token para solicitudes"  
  "email": "Email del usuario"  
  "name": "Nombre del restaurante"  
  "address": { "lat": Punto, "long": Punto, "direction": "Descripción de la ubicación" }  
  Opcional "number": teléfono del restaurante  
  Opcional "webPage": "Página web"  
  Opcional "foods": Array de comidas  
  Opcional "schedules": [{ "day": String, "open": Date, "close": Date }]  
}
```

Respuesta:

```
{  
  "token": "Token de respuesta"  
  "status": "Mensaje según lo sucedido"  
  "id": "id del restaurante"  
}
```

4.3.2. Registro de usuario: /updateRestaurant

Envío:

```
{  
  "token": "Token para solicitudes"  
  "email": "Email del usuario"  
  "id": "id del restaurante"  
  Opcional "name": teléfono del restaurante  
  Opcional "number": teléfono del restaurante  
  Opcional "webPage": "Página web"  
  Opcional "foods": Array de comidas  
  Opcional "schedules": [{"day": String, "open": Date, "close": Date}]  
}
```

Respuesta:

```
{  
  "token": "Token de respuesta"  
  "status": "Mensaje según lo sucedido"  
}
```

4.3.3. Registro de usuario: /createOpinion

Envío:

```
{  
  "token": "Token para solicitudes"  
  "email": "Email del usuario"
```

```
“id”: “id del restaurante”
“opinion”: { “qualification”: número, “price”: número, “date”: fecha, opcional “comment”: “Comentario del usuario” }
}
```

Respuesta:

```
{
“token”: “Token de respuesta”
“status”: “Mensaje según lo sucedido”
}
```

4.3.4. Registro de usuario: /getOpinions

Envío:

```
{
“token”: “Token para solicitudes”
“email”: “Email del usuario”
“id”: “id del restaurante”
}
```

Respuesta:

```
{
“token”: “Token de respuesta”
“opinions”: Array de opiniones
“status”: “Mensaje según lo sucedido”
}
```

4.3.5. Registro de usuario: /addPhoto

Recordar que varios archivos se pueden asociar a una llave(“photos”) simultáneamente, si no el servidor no obtendrá las imágenes.

Envío:

```
{
“token”: “Token para solicitudes”
“email”: “Email del usuario”
}
```

```
“id”: “id del restaurante”
“photos”: imágenes del restaurante (máximo 5)
}
```

Respuesta:

```
{
“token”: “Token de respuesta”
“status”: “Mensaje según lo sucedido”
}
```

4.3.6. Registro de usuario: /getPhotos

Envío:

```
{
“token”: “Token para solicitudes”
“email”: “Email del usuario”
“id”: “id del restaurante”
}
```

Respuesta:

```
{
“token”: “Token de respuesta”
“photos”: Array con la información binaria de las fotos
“status”: “Mensaje según lo sucedido”
}
```

4.3.7. Registro de usuario: /getRestaurants

Envío:

```
{
“token”: “Token para solicitudes”
“email”: “Email del usuario”
“filter”: json con los filtros (indicados más adelante)
}
```

Formato Json con los filtros

Envío:

```
{  
  Opcional "price": { "min": número, "max": número}  
  Opcional "qualification": { "min": número, "max": número}  
  Opcional "location": { "lat": número, "long": número, "distance": número}  
  Opcional "foods": Array de comidas permitidas  
}
```

Respuesta:

```
{  
  "token": "Token de repuesta"  
  "restaurants": Array con los restaurantes  
  "status": "Mensaje según lo sucedido"  
}
```

5. Interacción con sistemas externos

Para que se tenga un correcto funcionamiento de la aplicación desarrollada se requiere de tres sistemas independientes. El primero de ellos es el **API de Google Maps**, esta es utilizada para desplegar los marcadores de los restaurantes para que el usuario pueda de una manera visual buscar y seleccionar los restaurantes de interés. Por otro lado, se tiene el **API desarrollada para esta aplicación**, esta se encarga de responder todas las solicitudes de creación, modificación y consulta de los datos presentes en la base de datos, lo que nos lleva al tercer elemento requerido en esta aplicación. Este es la base de datos desarrollada en **MongoDB**.