UNIVERSIDAD DEL BIO-BIO

FACULTAD DE CIENCIAS EMPRESARIALES

DEPARTAMENTO DE SISTEMAS DE INFORMACION

UNIVERSIDAD DEL BÍO-BÍO

# "Re-construcción y mejora de plataforma thegoldenpages.info"

Proyecto de Software Aplicado presentado en conformidad a los requisitos para obtener el título de Ingeniero de Ejecución en Computación e Informática.

**Alumno:**
Felipe Adolfo Gatica Cea

**Profesor Guía:**
Patricio Galdames

Concepción, Agosto 2016

# Índice General

# Índice Tablas

# Índice Figuras

# 1 INTRODUCCIÓN

En el mundo en que vivimos actualmente hemos olvidado nuestra relación con el medio ambiente y sus demás seres vivientes con los que cohabitamos esta tierra. Con nuestro estilo de vida contaminamos, matamos y nos envenenamos casi sin saber que lo estamos haciendo y la sociedad hace un trabajo perfecto manteniéndonos de esa manera pensando que es lo normal y que “es imposible que lo que yo haga dañe tanto a nuestro planeta como a otros”.

La forma más propicia para cambiar esta situación es la educación y el ejemplo. Para esto las aplicaciones informáticas pueden cumplir un rol fundamental por su probada capacidad de influenciar en los procesos de las personas, como lo hemos visto con Facebook, Instagram o el mismo Google entre otros muchos ejemplos.

La revolución de la cuchara es una ONG[1] que trabaja para desarrollar alternativas de formación y difusión de información en conciencia ambiental, educación ante el consumo y promoción de hábitos de vida saludable. Trabaja en defensa de las personas, los animales y el medio ambiente.

En este marco esta ONG ha desarrollado la campaña “Pasaporte OKI”; documento físico de fantasía que acredita a quien lo posee como un "ciudadano del mundo", sin fronteras territoriales ni políticas, sin sesgos sociales, raciales e incluso de especie pues se considera a los animales también como ciudadanos con sus propios derechos. Además se invita a empresas y negocios pequeños a que se sumen a la campaña ofreciendo un descuento al portador del Pasaporte OKI. Para conseguir que estas empresas se animen a participar, la ONG los promociona dentro de sus plataformas virtuales. La más importante de éstas se llama las "Páginas Doradas" o “Golden Pages”; directorio de dichas empresas y negocios.

La plataforma virtual “The Golden Pages” o “Paginas Doradas” presenta una serie de problemáticas que limita su expansión y aceptación, tanto a las empresas que apoyan o contemplan apoyar como por las personas que participan de la campaña.
A continuación se encuentran las problemáticas observadas y/o indicadas por el cliente:

1. Interfaz poco amigable y difícil navegación.
2. La interfaz no se adapta al dispositivo usado.
3. La plataforma se siente lenta esto puede producir frustración y rechazo de los usuarios.
4. Los usuarios no son informados de ofertas o promociones.
5. Los dueños de negocios no pueden conocer las opiniones de sus clientes.
6. La Plataforma no está preparada para un aumento exponencial de usuarios o en otras palabras es difícilmente escalable.
7. La plataforma no cuenta con distintos tipos de usuarios.
8. El sistema de gestión no cuenta con información de la frecuencia de uso de la plataforma o problemas ocurridos a los usuarios.
9. El sistema actual no facilita su mantención ni menos su actualización, debido a que no se emplearon criterios de ingeniería de software en su desarrollo. Esta es una problemática importante ya que los desarrolladores son usualmente voluntarios que no permanecen por mucho tiempo.

De acuerdo a lo anterior, resulta vital la renovación completa de su actual plataforma para aumentar la aceptación de ambas partes comprometidas y mayor facilidad de exploración por parte de los usuarios.

Las siguientes son las soluciones propuestas por el autor de este proyeto de software. Cada punto es correlativo a un problema señalado anteriormente:

1. Rediseño de interfaz avalado con una prueba de usabilidad.
2. Usar Bootstrap para generar una interfaz responsiva.
3. Implementar compensación de latencias disponible usando Meteor para mejorar la sensación de fluidez de la aplicación.
4. Diseñar un sistema de "favoritos" que permita recibir alertas y/o notificaciones.
5. Diseñar e implementar un sistema de comentarios y/o valoraciones.
6. Implementar MongoDB para mejorar los tiempos de respuesta y posibilitar las escalabilidad horizontal.
7. Implementar distintos tipos de usuarios con sus respectivos roles y permisos.
8. Implementar sistema de informes, indicadores y/o estadísticas además de sección para recibir mensajes categorizados de los usuarios.
9. Utilizar Meteor y Blaze para desarrollar un código comprensible, reutilizable y mantenible.

Este proyecto de software estudia las mejoras que representan tanto las buenas prácticas como las nuevas tecnologías para aumentar la expansión y aceptación de dicha plataforma. Y presenta la implementación de dichas prácticas y tecnologías.

# 2 DEFINICIÓN PROYECTO

## 2.1 Objetivos del proyecto

### 2.1.1 Objetivo general

Potenciar (Reconstruir y adicionar características) la actual plataforma "The Golden Pages" en la difusión de una vida sana, la protección del medio ambiente y los animales a través de descuentos y promociones ofrecidos por empresas y negocios pequeños adheridos a la campaña del "Pasaporte OKI".

### 2.1.2 Objetivos específicos

- Investigar nuevas tecnologías y técnicas que supongan una mejora a las utilizadas actualmente en el mercado informático.
    - Formas de almacenar datos que supongan una mejora al tiempo de respuesta de una aplicación web.
    - Herramientas que mejoren la mantenibilidad de una aplicación web.
- Diseñar una nueva interfaz web para el sitio "The Golden Pages" bajo criterios de:
    - Usabilidad
    - Tiempo de Respuesta
    - Navegabilidad
- Diseñar plataforma de gestión para el sitio "The Golden Pages" que facilite la incorporación y gestión de servicios por parte de sus propios actores, como:
    - Promociones
    - Noticias
    - Favoritos
    - Calificaciones
- Diseñar una plataforma que facilite el seguimiento de los diversos actores que participan del sitio "The Golden Pages", con herramientas como:
    - Estadísticas tanto para administradores como para cada dueño de negocio.
    - Notificaciones de errores reportados.
- Verificar la calidad del producto final.

## 2.2 Ambiente de Ingeniería de Software

### 2.2.1 Metodología de desarrollo

Para desarrollar este proyecto se ha elegido el método ágil, más específicamente el SCRUM. Esta metodología se caracteriza por:

- Adoptar una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.
- Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos empleados.
- Solapamiento de las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o en cascada.

### 2.2.2 Metodología Ágil

Según el Manifesto para el desarrollo ágil de software[2] la valorización de los procesos que involucran el desarrollo se define de la siguiente manera:

- Los individuos y las interacciones por sobre los procesos y herramientas.
- Software funcional por sobre una documentación exhaustiva.
- Colaboración del cliente por sobre la negociación de contrato.
- Responder al cambio por sobre seguir un plan.

Bajo el alero de esta metodología se irán entregando "Incrementos" funcionales, que corresponden a versiones del sistema abocados a una funcionalidad o un grupo de ellas, de manera específica.

### 2.2.3 Herramientas de apoyo al desarrollo de software

Para la codificación e implementación del proyecto se usarán las siguientes herramientas:

- **Sublime Text 3**: Sofisticado editor de texto para código que permite snippets o pequeños trozos código reutilizable para **agilizar la escritura de aplicaciones**. Además, proporciona plugins y utilidades específicas para desarrolladores como **revisión de errores, selección e inserción múltiple de texto, etc.**
- **Meteor** es una plataforma **OpenSource** para crear aplicaciones web en tiempo real construida sobre Node.js. Dicha tecnología proporciona **reactividad** y **compensación de latencias** a la aplicación. Este framework será explicado más adelante para comprender mejor su apoyo en la solución de los problemas presentados.
- **MongoDB** es un sistema de **base de datos NoSQL** orientado a documentos, desarrollado bajo el concepto **OpenSource**. Esta tecnología seria usada para proporcionar **escalabilidad** a la aplicación.
- **NodeJS** es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor. **Requerido por Meteor y MongoDB.**
- **Blaze** es una librería construida dentro de Meteor para construir **interfaces reactivas, reutilizables y mantenibles.** Puede ser reemplazada por AngularJS o React.
- **Bootstrap 3**, es un **framework** originalmente creado por Twitter, que permite crear **interfaces web responsivas con CSS y JavaScript**.

## 2.3 Planificación del Proyecto

Ver Anexo 1: Carta Gantt

## 2.4 Definiciones, Siglas y Abreviaciones

**Amigo**: Negocio adherido a la campaña pero que aún vende productos de origen animal y se encuentra en proceso de concientización.

**BSON**: Representación binaria de estructuras de datos y mapas. Sus siglas significan Binary JSON. A diferencia de JSON, BSON está diseñado para tener un mejor almacenamiento y eficiencia.

**Embajador**: Poseedor de un "Pasaporte OKI".

**Framework**: En el ámbito de desarrollo de software, un framework o infraestructura, es una estructura conceptual que cuenta normalmente con módulos de software que sirven como base para la organización y desarrollo de un software.

**JSON**: JavaScript Object Notation es un formato de texto ligero para el intercambio de datos.

**Ministro**: Encargado de invitar negocios a adherirse a la campaña.

**Ministerio**: Negocio adherido a la campaña que no vende productos de origen animal.

**NoSQL**: Una clase de sistemas de gestión de bases de datos que difieren del modelo relacional.

**OpenSource:** Filosofía de desarrollo de software donde el código fuente es de dominio público, permitiendo a sus usuarios hacer cambios y aportes libremente.

# 3 MARCO TEÓRICO

En este capítulo introduciremos los conceptos y/o tecnologías investigadas para ser usadas en el desarrollo de este proyecto. Expondremos su utilidad y necesidad para dar solución a las problemáticas enfrentadas.

## 3.1 NoSQL

### 3.1.1 Definición

NoSQL o Not Only SQL es la terminología usada para referirse a toda base de datos y almacenes de datos que no utilizan el modelo relacional. Son usadas mayormente para almacenar grandes tamaños de datos con velocidades de respuesta superiores a las del modelo tradicional. NoSQL quiere decir que el lenguaje de consultas no necesariamente es el SQL.

Los sistemas NoSQL tienen una gran aplicación en la Web2.0 donde es necesaria la escalabilidad masiva, la computación paralela y los sistemas distribuidos. Este punto fue validado por Google cuando publico un paper llamado Bigtable: A Distributed Storage System for Structured Data[4] donde explica este nuevo enfoque y su propuesta para solucionar los problemas presentes en la pila de aplicaciones.

En años posteriores a dicha publicación, compañías como Facebook, Netflix, Yahoo, eBay, Hulu e IBM han adoptado los conceptos de NoSQL

### 3.1.2 Clasificación

Las bases de datos NoSQL pueden clasificarse de acuerdo a la forma en la que guardan los datos.

**Tabla 1 Clasificación de bases de datos NoSQL**

| Categoría | Descripción | Nombres de bases de datos |
|---|---|---|
| **Arreglos** | Bases de datos que hacen uso de arreglos multidimensionales para el almacenamiento y manejo de datos. | SciDB |
| **Clave-valor** | Almacenan los datos usando claves que identifican un valor específico. Los pares clave-valor son usados frecuentemente en las tablas hash, tablas de consulta y archivos de configuración. | BigTable, Cassandra, Dynamo, HBase, Hibari, Hypertable, MongoDB, Redis, Riak |
| **Documentales** | Los datos son almacenados como documentos que difieren en implementación por cada base de datos. Algunos ejemplos son XML, YAML, JSON y BSON. | BaseX, CouchDB, djondb, eXist, IBM Lotus, Jackrabbit,MongoDB, OrientDB, RavenDB, SimpleDB, Terrastore |

| | | |
|---|---|---|
| **Grafos** | Representan los datos usando nodos y aristas que representan relaciones, de esta manera se puede usar teoría de grafos para recorrer la base de datos. | AllegroGraph, DEX/Sparksee, FlockDB, HyperGraphDB, InfiniteGraph , InfoGrid, Neo4j, Sone GraphDB |
| **Multivalor** | Los datos pueden estar representados por varios valores al mismo tiempo al contrario de una única clave primaria en las bases de datos relacionales. | Caché InterSystems, Extensible storage engine, Jbase, OpenInsight, OpenQM, Reality, Rocket D3 DBMS, Rocket mvBase DBMS, Rocket U2 Unidata, Rocket U2 Universe |
| **Objeto** | Los datos se almacenan como objetos como lo haría cualquier lenguaje de programación orientado a objetos. | db4o, GemStone S, Objectivity/Db , Realm.io,Zope Object Database |
| **Tabular** | Son bases de datos "en memoria" usadas típicamente para análisis. | BigTable, HBase, Hypertable, LevelDB |

### 3.1.3 Características

En este apartado analizaremos las características de este tipo de bases de datos, enfocándonos específicamente en el impacto que tendrían en este proyecto. Otras características que no sean de nuestro interés no serán incluidas.

#### 3.1.3.1 Ausencia de esquema

En un sistema NoSQL un esquema de base de datos no es requerido, proporcionando libertad para almacenar la información. Si es necesario almacenar la información con otra estructura diferente en un lanzamiento posterior, simplemente se envía a la base de datos y esta se guarda sin más dificultad, no importando la estructura de los documentos previos.

Esta característica agiliza el tiempo de desarrollo al tener más flexibilidad para realizar cambios en la estructura de la base de datos. En una base de datos relacional, la obligación de utilizar esquemas rígidos puede decantar en tiempos de desarrollo más largos, más costosos y más complejos. Por otro lado la ausencia total de esquema puede llevar a una base de datos a tener una pobre calidad de datos y datos de poco valor. Es por ello que hemos sumado a Meteor, detallado más adelante, que nos permite utilizar esquemas solo cuando se necesite, un acercamiento más conveniente para este proyecto.

#### 3.1.3.2 No Relacional

Las bases de datos NoSQL no tienen el concepto de relaciones entre sus registros. En vez de esto se des normalizan los datos. Esto quiere decir que algunos datos pueden estar duplicados en varios registros.

Entonces existen dos ventajas en almacenar datos múltiples veces:

- **Almacenamiento y recuperación de los datos:** Solo escribir y leer un único registro con todos los datos necesarios.
- **Consultas veloces:** Una base de datos relacional une información al agregar restricciones a lo largo de las tablas en tiempo de consulta. Esto puede requerir que el motor de base de datos evalúe muchas tablas. Mientras más restricciones más se reduce el tiempo respuesta para dicha consulta. En las bases de datos NoSQL, toda la información que necesita evaluar se encuentra en un único documento. Por lo tanto, puede rápidamente determinar la lista de documentos coincidentes.

#### 3.1.3.3 Estructura distribuida

Particionado de los datos hacia bases de datos locales. Permite procesar consultas localmente o globalmente entre varias localidades.

Esta característica nos permitiría distribuir los datos en cada país en el que esta campaña se desarrolla. Estos países se listan a continuación: Alemania, Argentina, Bolivia, Brasil, Bulgaria, Chile, Colombia, Costa Rica, Ecuador, España, Estados Unidos, Filipinas, Guatemala, Hungría, Italia, México, Panamá, Paraguay, Países Bajos, Perú, Suecia, Suiza. India, Tailandia, Uruguay y Venezuela.

Actualmente esta función no ha sido solicitada para ser implementada inmediatamente pero si es una característica deseada.

#### 3.1.3.4 Escalabilidad horizontal

Capacidad de añadir nodos al sistema agregando capacidad de procesamiento y almacenamiento.

Esta característica le permite a la ONG agregar nodos de bajo costo en contraposición a cambiar el sistema a una maquina más poderosa y costosa.

#### 3.1.3.5 Tolerancia a fallos y redundancia

Este tipo de sistemas tienen la capacidad de proporcionar redundancia de datos y gran tolerancia a fallos a través de replicación y estructura distribuida.

La replicación será explicada más adelante en la sección MongoDB.

#### 3.1.3.6 Consistencia eventual

Los sistemas de bases de datos NoSQL soportan esta característica, aunque no es obligatoria ya que pueden configurarse para aplicar otro tipo de consistencia. Consistencia eventual se refiere a la implementación de mecanismos de consistencia flexibles que permitan mantener un sistema distribuido y con una alta disponibilidad.

Dicha característica puede verse como una desventaja, aunque ello depende de la aplicación a desarrollarse.

Este punto se puede explicar usando el teorema CAP[3]:

**Consistencia**: Todos los clientes de la base de datos ven la misma información, inclusive con modificaciones concurrentes.
**Disponibilidad**: Todos los clientes de la base de datos son capaces de acceder alguna versión de la información.
**Tolerancia a la Partición**: La base de datos puede ser divida sobre múltiples servidores.


*Ilustración 1 Teorema CAP [3]*

En este grafico se puede ver que MongoDB, la base de datos a utilizar, se encuentra entre consistencia y tolerancia a la partición debido a que, por defecto, esta solamente permite la lectura desde un primario a menos que opcionalmente se habilite la lectura desde secundarios donde se vuelve eventualmente consistente. Esto será explicado más adelante en 3.2.2.3 Replicación. La consistencia eventual solo es un problema si los datos que manejamos son especialmente sensibles como los datos manejados por un banco. Por otro lado, si una notificación llega a destiempo, no es un problema para este proyecto.

## 3.2 MongoDB

### 3.2.1 Definición

Su nombre proviene de la palabra inglesa "hu**mongo**us" que significa enorme. Es un sistema de bases de datos NoSQL que utiliza documentos para almacenar sus datos y es desarrollado bajo el concepto de código abierto.

El formato de sus "documentos" es el BSON una forma de JSON que facilita la integración de los datos en ciertas aplicaciones.

No requiere de esquemas, es escalable y de alto rendimiento.

### 3.2.2 Características

Al igual que en el apartado de características de un sistema NoSQL nos avocaremos principalmente a las características que nos interesan o que puedan representar una desventaja al proyecto de software.

#### 3.2.2.1 Consultas Ad hoc

MongoDB soporta búsqueda por campos, consultas de rangos y expresiones regulares. Puede por lo tanto devolver una porción del registro a pedido, así como una función JavaScript definida por el usuario.

Como se puede ver, MongoDB no es muy diferente a una base de datos como MySQL en este caso por lo que usar este tipo de bases de datos no representa un problema.

#### 3.2.2.2 Indexación

Capacidad de crear índices a partir de cualquier campo, al igual que es posible hacer índices secundarios. Funciona muy parecido a una base de datos relacional.

#### 3.2.2.3 Replicación

MongoDB puede determinar grupos de replicación de tipo primario y secundario denominados replica set. El primario puede ejecutar comandos de lectura y escritura mientras que los secundarios replican los datos del primario y solo pueden ser utilizados para lectura o copia de seguridad. El grupo secundario puede cambiar de primario si este ha dejado de responder.

La principal ventaja de esta característica es la alta disponibilidad de los datos al cambiar de primario ante un eventual problema físico.

#### 3.2.2.4 Balanceo de carga

MongoDB permite escalar horizontalmente usando el concepto de "shard". Esto permite dividir los datos en rangos basados en un clave de sharding y distribuirlos a través de múltiples shard.

#### 3.2.2.5 Agregación

MongoDB permite operaciones de agregación a través de un framework proporcionando funciones similares a las encontradas en SQL tales como AVG(), MIN(), MAX(), etc. Además, proporciona una función llamada MapReduce que puede ser utilizada para el procesamiento paralelo de datos por lotes y operaciones complejas de agregación a lo largo de varios servidores.

MapReduce está compuesto de varias funciones en JavaScript y ejecutadas en serie:

1. Map: Cada nodo recolecta los documentos requeridos y transforma los datos en un Mapa de tipo clave-valor.
2. Shuffle: Agrupa los valores por clave.
3. Reduce: Los nodos procesan cada grupo de datos, por clave, en paralelo

### 3.2.3 Conceptos básicos

A continuación, se presentan y explican brevemente los conceptos básicos de esta base de datos de manera que podamos comprender su funcionamiento general y despejar dudas respecto a su efectividad en el almacenamiento de datos para esta aplicación específica.

#### 3.2.3.1 Documentos

El formato para los documentos usados en MongoDB es el BSON, esta es una versión mejorada del formato JSON.

Cada documento es similar a una tupla en una base de datos relacional y almacena los datos en un conjunto de pares clave-valor.

```
{
    nombre: "Saldaña",
    direccion: {
        calle: "Caupolicán 889",
        ciudad: "Concepción"
    },
    productos: [
        "nueces",
        "maní",
        "pasas"
    ],
}
```

*Ilustración 2 Ejemplo de documento*

En un documento, lo que sería una columna ahora es un atributo. Dicho atributo puede ser de cualquier tipo de dato incluyendo otro documento, lo que toma el nombre de documento anidado. Para acceder a algún atributo dentro de un documento anidado se utiliza el carácter punto, por ejemplo "*dirección.calle*" en la Ilustración 2.

Cada documento posee un atributo _id que lo identifica de manera única y es analógica a la clave primaria de las bases de datos relacionales. Si el atributo _id no es especificado en la creación del documento MongoDB genera uno automáticamente.

```
{
    _id: ObjectId("4efa8d2b7d284dad101e4bc7"),
    nombre: "Saldaña",
    direccion: {
        calle: "Caupolicán 889",
        ciudad: "Concepción"
    },
    productos: [
        "nueces",
        "maní",
        "pasas"
    ],
}
```

*Ilustración 3 Documento ejemplo con _id autogenerado*

El tamaño máximo de un documento es de 16 Megabytes a menos que se implemente la funcionalidad de GridFS donde no existe un límite teórico y solo se ve acotado por el espacio de almacenamiento físico del cluster en el que se aloja.

#### 3.2.3.2 Colecciones

Las colecciones son almacenes de documentos, podemos compararlas a las tablas de un modelo relacional. En una colección podemos realizar tareas como leer y actualizar, eliminar documentos, también podemos crea índices, realizar operaciones de mantenimiento y otras funcionalidades útiles.

#### 3.2.3.3 Caché de MongoDB

Para aumentar la velocidad de acceso a los datos MongoDB hace uso de la memoria del sistema para mantener porciones de datos que son consultados, si los datos no son consultados permanecen en la memoria secundaria. Antes de manipular o acceder a los datos estos deben estar en memoria.

## 3.3 Meteor.JS

### 3.3.1 Definición

Meteor es un framework o conjunto de herramientas para desarrollar aplicaciones web en tiempo real. Meteor se encuentre entre la base de datos y la interfaz de usuario y se encarga de que ambas partes estén sincronizadas.

Meteor está construida sobre Node.js, por lo tanto, utiliza JavaScript tanto en el cliente como el servidor y es capaz de compartir código entre ambos entornos.

### 3.3.2 Características principales

A continuación, se presentan las características de Meteor que nos convencieron de utilizar este framework para el desarrollo de esta aplicación.

- Paquetes
- Renderizado reactivo
- Compensación de latencias
- Sistema de Renderizado Blaze
- Código Javascript Isomorfo

Estas características serán profundizadas en los próximos puntos:

#### 3.3.2.1 Paquetes

Además de las características incluidas por defecto, Meteor te ofrece la posibilidad de instalar paquetes desarrollados por la comunidad o privados que extenderán las funcionalidades de la aplicación sin muchos problemas. De esta manera podemos integrar herramientas como Bootstrap, JQuery, entre otros, en cosa de segundos gracias a su línea de comandos.

```
$ meteor add twbs:bootstrap
```

*Ilustración 4 Ejemplo línea de comandos para instalar módulos*

Paquetes utilizados:

- **CollectionFS:** Es un conjunto de paquetes para Meteor que juntos proveen una solución completa para el manejo de archivos incluyendo subida, descarga, almacenamiento, sincronización, manipulación y copia.
- **Autoform:** Provee de la habilidad de autogenerar formularios en HTML5 basados en un esquema de base de datos. Incluye componentes de interfaz de usuario por defecto, como cajas de arrastre de archivos, así como la posibilidad de agrega más elementos por medio de paquetes compatibles.
- **Universe select:** Paquete compatible con Autoform que provee un complemento de tipo "select" para la interfaz de usuario.
- **Autoform tags:** Paquete compatible con Autoform que provee un complemento de interfaz que permite la agregación de etiquetas.
- **Collection2:** Permite adjuntar un esquema de base de datos a una colección de MongoDB. Valida automáticamente basándose en el esquema cuando se inserta o actualiza desde el cliente y el servidor.
- **Simple-Schema:** Paquete necesario para utilizar Collection2 que nos permite crear esquemas de bases de datos y reglas de validación.
- **Bert:** Sistema de alertas para el cliente.
- **Bootstrap:** Paquete que integra el conocido framework de Twitter. Permite crear interfaces responsivas.

Esta característica nos permite avanzar aún más rápido y reducir costos de implementación de nuevas funciones a la aplicación en el futuro.

#### 3.3.2.2 Renderizado reactivo

Esta característica implica que los elementos en la aplicación se actualizan de forma automática ante cualquier cambio tanto en la base de datos como en la interfaz de usuario. Esto quiere decir que, si un usuario realiza, por ejemplo, una inserción en la base de datos, todos los que tengan acceso a esos datos verán reflejado el cambio en su interfaz gráfica mostrando el nuevo elemento automáticamente sin necesidad de programar ningún tipo de sincronización o callback. Esto se aplica igualmente si se realiza un cambio a la interfaz en tiempo de ejecución este se verá reflejado automáticamente para cada usuario que esté utilizando la aplicación sin necesidad de refrescar la página o reiniciar el servidor.

Esta característica dará al usuario la sensación de estar usando una aplicación de escritorio lo cual es muy deseado en aplicaciones web.

#### 3.3.2.3 Compensación de latencia

Meteor permite implementar la "compensación de latencia" el cual mejora el tiempo de respuesta de la aplicación. Dicho concepto es introducido por primera vez gracias a Meteor.

En una aplicación tradicional, cuando un usuario realiza alguna acción en la aplicación este tiene que esperar hasta que el servidor procese su petición para ver los resultados reflejados en la interfaz de usuario. Gracias a la compensación de latencia el usuario ya no necesita esperar.

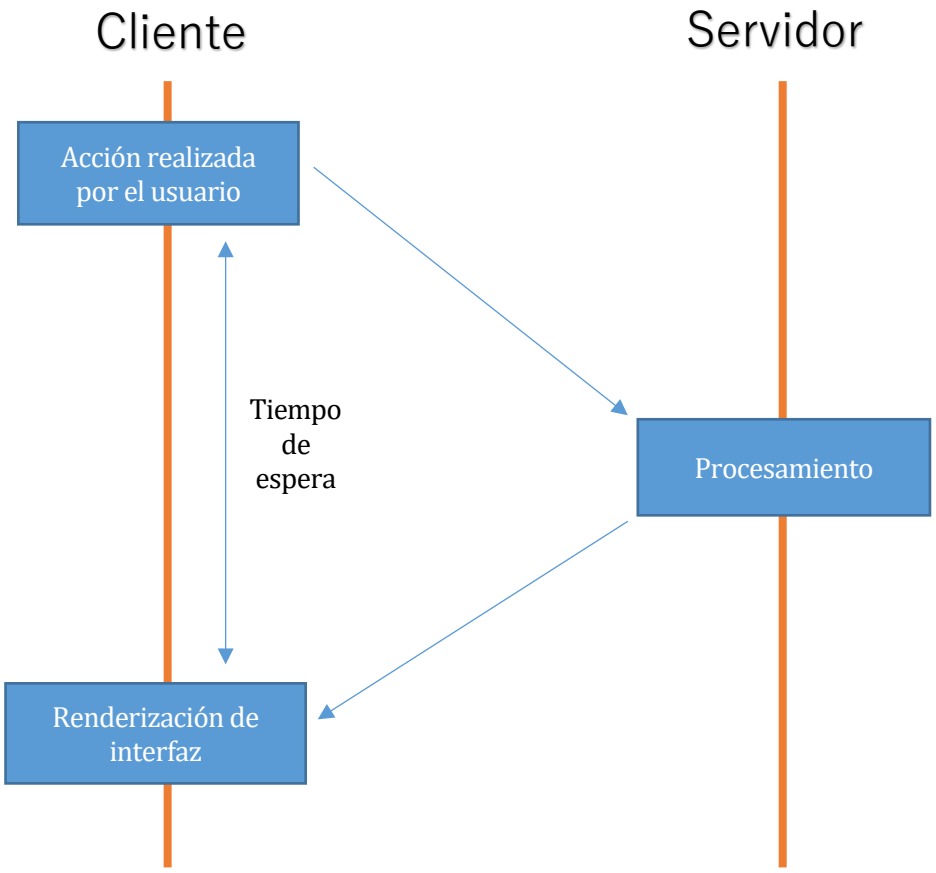



*Ilustración 5 Sin compensación de latencia. [5]*

En una aplicación con esta funcionalidad implementada, lo que ocurre es que, como normalmente ocurre, el sistema enviará la petición al servidor, pero el cliente reflejara los cambios inmediatamente en la interfaz utilizando una base de datos local no persistente llamada Minimongo, luego al recibir la respuesta del servidor, el cliente mezclará los datos mediante un *merge*. Usualmente, los datos del servidor y el cliente serán iguales a menos que el servidor agregue algún dato extra en tiempo de inserción.

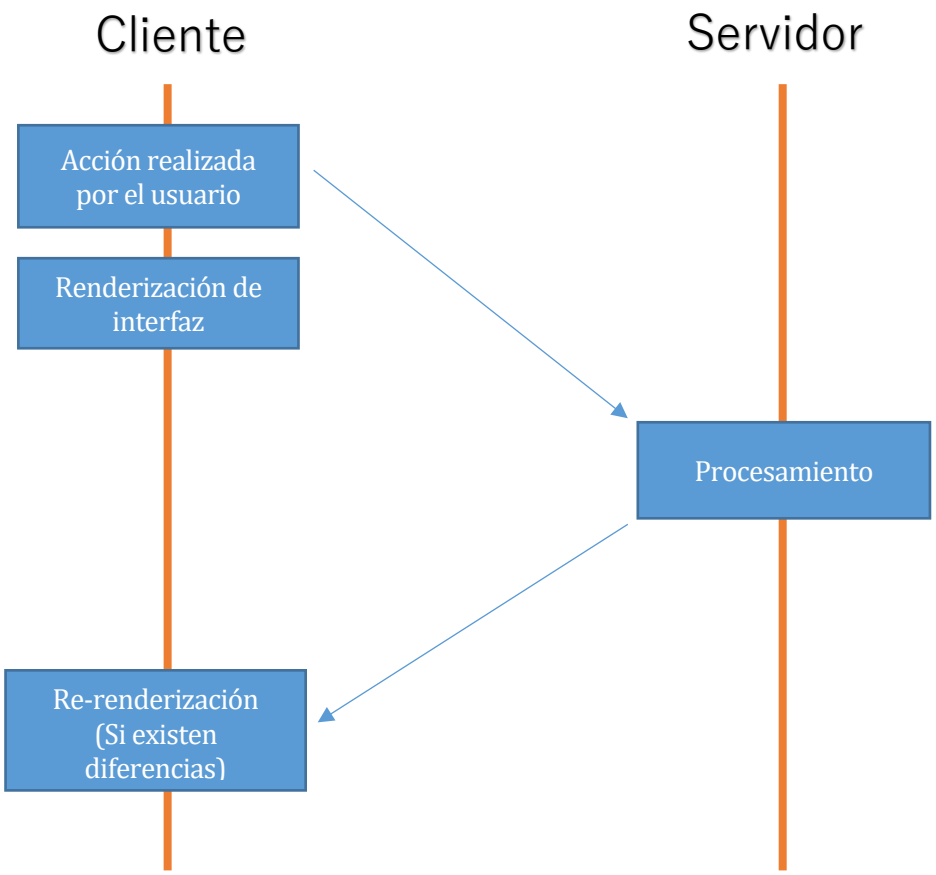



*Ilustración 6 Con compensación de latencia. [5]*

En caso de que el servidor rechace la petición, el cliente lo detectara y el usuario verá como su acción será revertida. Esto puede sorprender al usuario, pero existen formas de manejarlo haciendo uso de alertas u otros mecanismos.

Vale decir que esta característica debe ser implementada obligadamente por el desarrollador y puede ser ignorada en caso de necesitar esperar por una respuesta del servidor.

Esta característica proporciona una obvia mejora a la experiencia del usuario lo que ayudará a promocionar la campaña y el uso de la aplicación que es el objetivo principal de la plataforma "The Golden Pages"

#### 3.3.2.4 Sistema de Renderizado Blaze

Blaze es un sistema de renderizado desarrollado por el equipo detrás de Meteor que nos ayuda a desarrollar interfaces de usuario reactivas, usables y mantenibles.

Blaze implementa el concepto de plantillas o templates para definir trozos de código que definirán partes de la interfaz.

```
1  <template name="leaderboard">
2    <ol class="leaderboard">
3      {{#each players}}
4        {{> player}}
5      {{/each}}
6    </ol>
7  </template>
8
9  <template name="player">
10   <li class="player {{selected}}">
11     <span class="name">{{name}}</span>
12     <span class="score">{{score}}</span>
13   </li>
14 </template>
```

*Ilustración 7 Ejemplo Template*

Como se ve en la Ilustración 7, con Blaze podemos generar pequeños trozos de código que se pueden insertar en otras partes de nuestra aplicación a discreción del desarrollador, como se ve en la línea 4 donde se inserta la plantilla llamada *"player"*. Además de esto también podemos ver varias etiquetas como *{{name}}* y *{{score}}* en las líneas 11 y 12 las cuales se refieren a propiedades en el contexto de la plantilla actual, mientras que "*players"* y *"selected"* se refieren a funciones de tipo *helpers* definidas en JavaScript.

Otra característica especial de las plantillas es el hecho de poder usar operadores condicionales y lógicos en la interfaz, como podemos ver por la etiqueta *{{#each}}* en la línea 3, lo que quiere decir que por cada *"players"* devuelto el sistema Blaze insertara la plantilla *"player"*. Tambien podemos hacer uso de {{#if}}, {{#unless}}, {{#each .. in}}, entre otros operadores.

```
1  Template.leaderboard.helpers({
2    players: function () {
3      // Perform a reactive database query against minimongo
4      return Players.find({}, { sort: { score: -1, name: 1 } });
5    }
6  });
7  Template.player.events({
8    'click': function () {
9      // click on a player to select it
10     Session.set("selectedPlayer", this._id);
11   }
12 });
13 Template.player.helpers({
14   selected: function () {
15     return Session.equals("selectedPlayer", this._id) ? "selected" : '';
16   }
17 });
```

*Ilustración 8 Ejemplo Javascript de un Template*

La línea 4 en la Ilustración 8 devuelve los *"Players"* almacenados en la base de datos ordenados por *"score"* descendentemente y por *"name"* ascendentemente.

#### 3.3.2.5 Código Javascript Isomorfo

Meteor permite usar el mismo código en tanto en el front-end como en el back-end asi también como para móviles como para aplicaciones web. Previene a los desarrolladores de requerir instalar y configurar diferentes librerías, gestores de modulos, APIs, controladores y más.

Esto quita mucho tiempo de desarrollo al no tener que cambiar de contexto entre el lenguaje del servidor y el lenguaje de la aplicación.

# 4 PRODUCT BACKLOG

## 4.1 Alcances

Esta sección describe el product packlog del proyecto “The Golden Pages" cuyo objetivo principal es promover la vida sana, la protección de la naturaleza y los animales a través de una dieta y consumo concientes.

Este proyecto de software integrará nuevas tecnologías y conceptos usados actualmente por reconocidas empresas como Google, Facebook, Cisco, Ebay, Forbes en el caso de MongoDB; y Mazda, Ikea, Honeywell en el caso de Meteor solo por nombrar unos pocos. Estas empresas tienen en mente la velocidad de reacción, el manejo de cantidades enormes de datos en poco tiempo. Tomando esto en cuenta se ha decidido investigar estas tecnologías y finalmente usarlas para dar solución a los problemas encontrados y comentados por nuestro cliente.

Este software ayudará a mejorar la comunicación entre clientes y consumidores concientes, la propagación de la vida sana y la protección tanto del medio ambiente como de los animales por medio de un directorio de negocios/empresas que apoyen la ya mencionada campaña OKI. Permitirá a administradores conocer cómo funciona la plataforma a través de estadísticas y retroalimentación proveniente de los mismos usuarios como problemas, errores, quejas y recomendaciones. Todo esto en una interfaz fácil de entender y navegar, manteniendo colores que identifican la campaña y la ONG.

## 4.2 Objetivo del software

- Almacenar registros de los negocios/empresas asociados/as a la campaña OKI para mejorar su visibilidad hacia las personas participantes de la campaña.
- Permitir la gestión del contenido de los registros para posibilitar la constante mejora de la plataforma y mantener un dinamismo que atraiga al público.
- Notificar a los usuarios sobre las actualizaciones/promociones de sus negocios/empresas favoritas/as y así mantenerlos informados oportunamente.
- Entregar estadísticas útiles para administradores y dueños de negocios para facilitar la respuesta efectiva a la actividad en la plataforma logrando un mejor control sobre su funcionamiento.
- Manejar las diferentes vistas de la interfaz para mejorar su utilización en diferentes dispositivos.

## 4.3 Descripción Global del Producto

### 4.3.1 Interfaz de usuario

- Colores representativos de la campaña.
- Diseño minimalista.
- No más de seis resultados por página al efectuar una búsqueda.
- Debe ser posible cambiar de idioma entre Español, Inglés y Portugués.
- Debe cambiar de apariencia según tamaño de pantalla.

## 4.4 Requerimientos Específicos

### 4.4.1 Requerimientos Funcionales del sistema

A continuación, se detallan los requerimientos comentados por nuestro cliente, así como los observados por el equipo.

El orden en el que se detallan los requerimientos será por prioridad. Esta prioridad será usada como "Product Backlog" para el desarrollo en "Scrum".

No será detallado el responsable de cada desarrollo ya que el único participante será el autor de este proyecto de software.

**Tabla 2: Requerimientos funcionales**

| ID | Prioridad | Requerimiento | Descripción | Autoría |
|---|---|---|---|---|
| **REF-1** | Muy Alta | Generación de cuentas | Tanto el Administrador como los Ministros necesitan generar cuentas de usuarios.<br><br>Especificación:<br>1. Generar cuentas de tipo:<br>• Ministro<br>• Negocio<br>2. Datos necesarios:<br>• Correo<br>• Tipo de Cuenta<br>3. La contraseña debe ser autogenerada.<br>4. El correo y la contraseña pueden ser cambiados posteriormente.<br>5. Los datos de quien generó la cuenta deben ser almacenados | Alexis Ulloa |
| **REF-2** | Muy Alta | Registro de negocio | Se necesitan registrar los detalles de la empresa o negocio que se adhiere a la campaña.<br><br>Especificación:<br>1. Una cuenta de tipo negocio solo puede tener uno y solo un registro de negocio.<br>2. Los datos a ingresar son:<br>• Nombre<br>• Subtitulo<br>• Descripción (140 Caracteres)<br>• Fotos (1 min – 5 max)<br>• Descuento | Alexis Ulloa |

| | | | | |
|---|---|---|---|---|
| | | | • Categoría<br>• Dirección<br>• País<br>• Ciudad<br>• Página Web<br>• Horarios de Atención<br>• Lista de Productos<br>• Sucursales<br>3. Luego de finalizar el registro el negocio puede modificar los datos en cualquier momento. | |
| **REF-3** | Muy Alta | Búsqueda | Se necesita poder realizar una búsqueda filtrada por una serie de criterios:<br>• Texto<br>• Categoría<br>• País<br>• Ciudad<br><br>Especificación:<br>1. Se deben mostrar máximo 6 resultados por cada página. | Alexis Ulloa |
| **REF-4** | Muy Alta | Calificar Negocio | Existen dos tipos de calificaciones: Cucharas y Estrellas<br><br>Especificación:<br>1. Un embajador debe ser capaz de calificar a un negocio con estrellas y agregar un comentario.<br>2. Un Ministro debe ser capaz de calificar a un negocio con cucharas y agregar un comentario.<br>3. Las cucharitas deben llevar su explicación sobre su activismo por cada cucharita. | |

| | | | | |
|---|---|---|---|---|
| **REF-5** | Muy Alta | Catalogar Negocio | Tanto el Administrador como los Ministros pueden catalogar a un Negocio como "Amigo" o "Ministerio".<br><br>Especificación:<br><br>1. El usuario puede elegir el negocio a partir de una lista.<br>2. Esta lista muestra solo los negocios a cargo del embajador y todos los negocios en caso del Administrador.<br>3. A partir de esta lista el usuario puede seleccionar un negocio y cambiar varios campos, entre estos, la categoría de amigo o ministerio. | Alexis Ulloa |
| **REF-6** | Alta | Registro de Usuarios | El visitante común debe ser capaz de registrarse en la plataforma para acceder a servicios específicos.<br><br>Especificación:<br>1. Datos Necesarios:<br>• Correo<br>• Contraseña<br>2. Un usuario registrado de esta manera es clasificado automáticamente como "Embajador".<br>3. Luego de ser registrado el usuario puede modificar sus datos en cualquier momento. | Alexis Ulloa |
| **REF-7** | Alta | Agregar "Favoritos" | El Embajador debe ser capaz de agregar uno o más negocios a sus favoritos.<br><br>Especificación:<br>1. Esta acción debe poder ejecutarse desde la lista resultante en la búsqueda o dentro del perfil del negocio. | Alexis Ulloa |

| | | | | |
|---|---|---|---|---|
| **REF-8** | Alta | Ver Estadísticas e Informes | Tanto Administrador, Ministros y Negocios deben tener acceso a distintos tipos de estadísticas.<br><br>Especificación:<br>Las estadísticas a mostrar serán:<br>• Administrador:<br>1. Embajadores Activos (1 mes sin entrar se considera inactivo).<br>2. Negocios Activos (1 mes sin entrar se considera inactivo).<br>3. Visitas Mensuales a la plataforma.<br>4. Problemas notificados por los clientes (Nivel plataforma, ej: Errores, Bugs, etc).<br>• Ministros:<br>1. Actividad de Negocios (Visitas mensuales, último inicio de sesión).<br>2. Mostrar Negocios que no hayan iniciado sesión en un mes o más.<br>3. Problemas a nivel de usabilidad, ej: Desconocimiento de funciones, dudas, problemas físicos como falta de material de activismo.<br>• Negocios:<br>1. N° Favoritos<br>2. N° de Visitas Mensuales<br>3. Comentarios y calificaciones. | |

| | | | | |
|---|---|---|---|---|
| **REF-9** | Media | Registrar Promoción | Un negocio puede registrar una nueva promoción.<br><br>Especificación:<br>1. El negocio puede agregar una promoción indicando los siguientes detalles:<br>a. Titulo<br>b. Descripción<br>c. Fecha de termino<br>d. Descuento (opcional) | Alexis Ulloa |
| **REF-10** | Media | Desactivar/Eliminar Promoción | Un negocio puede desactivar/eliminar sus promociones.<br><br>Especificación:<br>1. Desactivar una promoción significa que la promoción queda inactiva pero aun presente en la base de datos.<br>2. Eliminar una promoción significa que se borrara de la base de datos permanentemente. | Alexis Ulloa |
| **REF-11** | Media | Re-activar promoción | Con un simple clic el negocio debe ser capaz de reactivar su promoción.<br><br>Especificación:<br>1. Al momento de reactivar una promoción se debe pedir una fecha de término de la promoción para no generar incongruencias. | |
| **REF-12** | Media | Gestionar Lista de Favoritos y sus servicios | El embajador debe ser capaz de ver una lista con sus favoritos y gestionarla. Lo mismo se aplica a los servicios asociados a estos favoritos.<br><br>Especificación:<br>1. El embajador debe ser capaz de Eliminar uno o más favoritos.<br>2. Debe ser capaz de desactivar/activar notificaciones. | Alexis Ulloa |

| | | | | |
|---|---|---|---|---|
| **REF-13** | Baja | Ceder Control sobre Negocio | Un Administrador puede ceder el control sobre un negocio a otro Ministro.<br><br>Especificación:<br><br>1. Al momento de quitar el control sobre un negocio el ministro debe ser notificado sobre los cambios. | Alexis Ulloa |

### 4.4.2 Requerimientos No Funcionales del Sistema

| ID | Requerimiento | Descripción |
|---|---|---|
| **RNF-1** | Simplicidad de uso | La navegación de la plataforma debe implicar la menor cantidad de clics posibles. |
| **RNF-2** | Diseño corporativo | La plataforma debe utilizar colores que representen la campaña. Estos colores son los de un pasaporte (Azul, amarillo, blanco) |
| **RNF-3** | Encriptación de datos | Los datos sensibles de los usuarios deben ser encriptados. |
| **RNF-4** | Tiempo de respuesta | El tiempo requerido para responder a una petición debe ser la menor máximo 3 segundo. |
| **RNF-5** | Portabilidad | La interfaz debe adaptarse para Móviles y Tabletas. |
| **RNF-6** | Escalabilidad | La plataforma permitir la agregación de potencia para permanecer fluida frente a un crecimiento continuo de usuarios. |
| **RNF-7** | Mantenibilidad | La plataforma debe ser desarrollada utilizando buenas prácticas que faciliten su entendimiento para desarrolladores futuros. |

# 5 ANÁLISIS

## 5.1 Casos de uso

A continuación, se incluirán los casos de uso previstos para la nueva plataforma y su explicación detallada

### 5.1.1 Diagrama Casos de Uso

A continuación, se detallarán los diferentes diagramas de casos de uso. Se mostrarán separadamente por cada tipo de usuario para mejorar su visualización y comprensión.

#### 5.1.1.1 Embajador

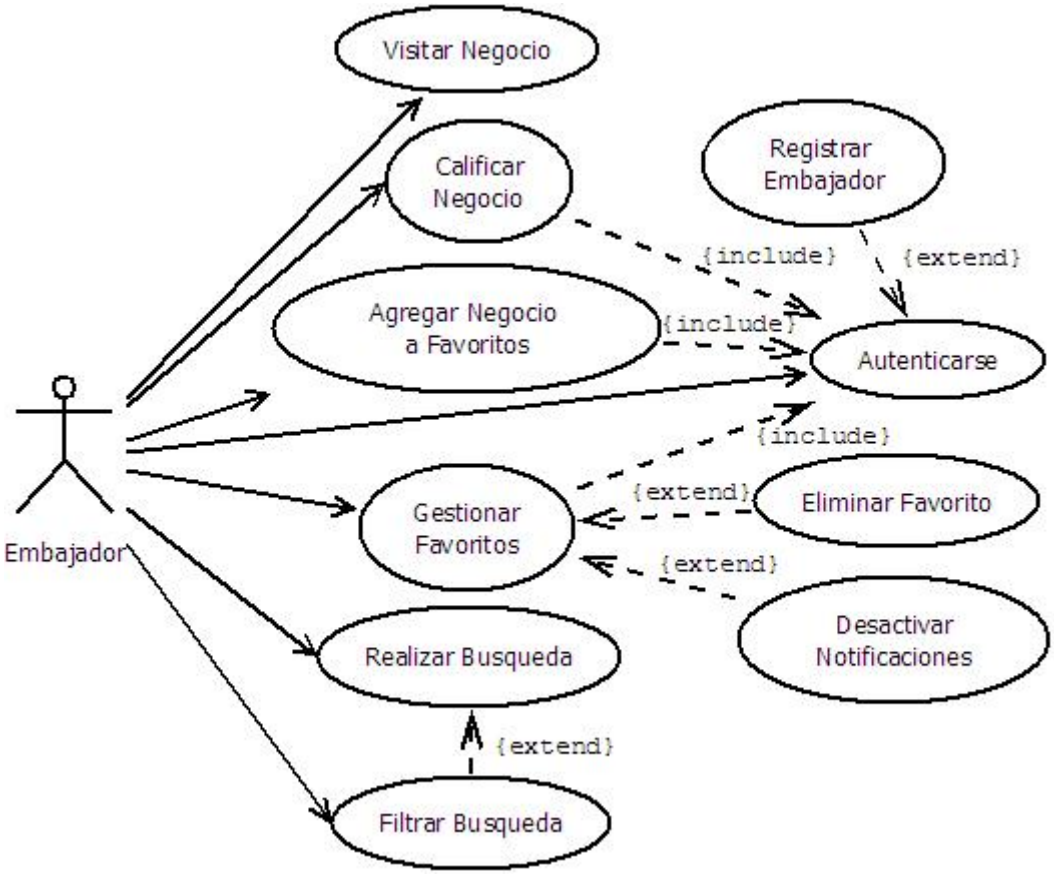



*Ilustración 9 Diagrama de Casos de Uso para un Embajador*

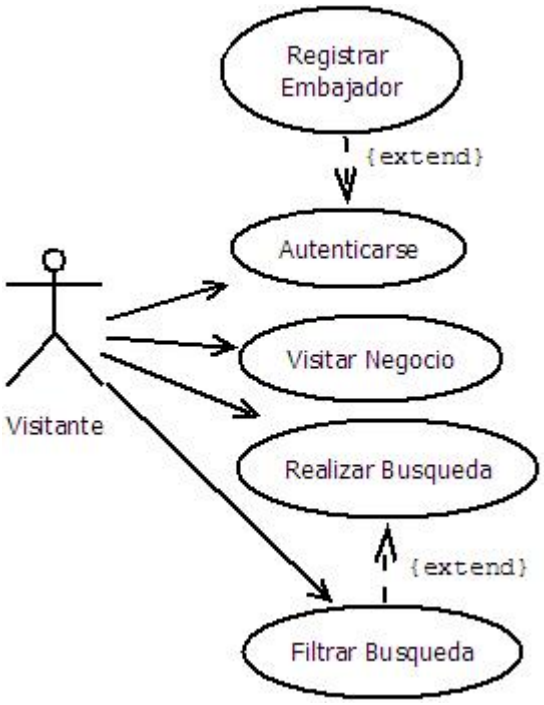



*Ilustración 10 Diagrama de Casos de Uso para un Visitante*

#### 5.1.1.2 Actores

- **Embajador:** Corresponde al usuario registrado en el sistema.
- **Visitante:** Corresponde a la persona que visita la aplicación sin estar registrado.

#### 5.1.1.3 Especificación de los Casos de Uso

**1. Caso de uso**: Autenticarse.
**Actores**: Administrador, Ministro, Negocio, Embajador, Visitante.
**Objetivo**: Permitir al usuario ingresar al sistema
**Precondiciones**: El sistema debe estar disponible.
**Post-condiciones**: El usuario entra al sistema.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El usuario ingresa su email y clave personal para ingresar al sistema con sus respectivos privilegios. | 2. Verifica los datos ingresados al sistema. |
| | 3. Si los datos ingresados son correctos se ingresa al sistema. |

Cursos Alternos:

- Ítem 1: El Visitante que no tiene una cuenta puede proceder al caso de uso nº 2.
- Ítem 3: Si los datos ingresados son incorrectos, el sistema despliega un mensaje de error.

**2. Caso de uso**: Registrar Embajador.
**Actores**: Visitante.
**Objetivo**: Permitir al visitante registrarse en el sistema
**Precondiciones**: El sistema debe estar disponible.
**Post-condiciones**: El visitante es registrado en el sistema y es autenticado automáticamente.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El visitante ingresa su email y clave personal para ser registrado. | 2. Verifica los datos ingresados al sistema. |
| | 3. Si los datos ingresados son correctos el visitante es registrado como embajador e ingresa al sistema automáticamente. |

Cursos Alternos:
- Ítem 3: Si los datos ingresados son incorrectos, el sistema despliega un mensaje de error.

**3. Caso de uso**: Realizar Búsqueda.
**Actores**: Administrador, Ministro, Negocio, Embajador, Visitante.
**Objetivo**: Permitir al usuario realizar una búsqueda de negocios.
**Precondiciones**: El sistema debe estar disponible.
**Post-condiciones**: El sistema debe desplegar los resultados de la búsqueda.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El usuario introduce texto en el campo de búsqueda. | 2. El sistema realiza la búsqueda intentando correlacionar el texto ingresado con el nombre de los negocios y con los productos indicados por los negocios. |
| | 3. El sistema despliega los resultados de la búsqueda. |

Cursos Alternos:
- Ítem 3: Si la búsqueda no devuelve ningún resultado será notificado por pantalla.

**4. Caso de uso**: Filtrar Búsqueda.
**Actores**: Administrador, Ministro, Negocio, Visitante, Embajador.
**Objetivo**: Permitir al usuario filtrar una búsqueda de negocios.
**Precondiciones**: El sistema debe estar disponible.
**Post-condiciones**: El sistema debe desplegar los resultados del filtro.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El usuario selecciona los argumentos para el filtrado:<br>a. País<br>b. Cuidad<br>c. Categoría | 2. El sistema filtra la búsqueda de acuerdo a los argumentos ingresados reactivamente. |
| | 3. El sistema despliega los resultados del filtrado. |

Cursos Alternos:
- Ítem 3: Si la búsqueda no devuelve ningún resultado será notificado por pantalla.

**5. Caso de uso:** Visitar Negocio.
**Actores**: Administrador, Ministro, Negocio, Embajador, Visitante.
**Objetivo**: Permitir al usuario revisar los detalles de un negocio.
**Precondiciones**: El sistema debe estar disponible.
**Post-condiciones**: El sistema debe desplegar los detalles del negocio.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El usuario selecciona un negocio desde la página principal o luego de haber realizado una búsqueda. | 2. El sistema despliega los detalles del negocio. |

**6. Caso de uso**: Agregar Negocio a Favoritos.
**Actores**: Administrador, Ministro, Negocio Normal, Embajador.
**Objetivo**: Permitir al usuario agregar un negocio a su lista de favoritos.
**Precondiciones**: El sistema debe estar disponible y haber ejecutado el caso de uso nº 5.
**Post-condiciones**: El sistema debe agregar el negocio seleccionado a la lista de favoritos del usuario.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El usuario hace clic sobre el botón “Agregar a Favoritos”. | 2. El sistema agrega el negocio a la lista de favoritos del usuario. |
| | 3. El sistema despliega una alerta sobre el resultado de la acción. |

Cursos Alternos:

- Ítem 2: Si el sistema no puede ejecutar la inserción de manera correcta alertara al usuario en el siguiente ítem.

**7. Caso de uso**: Gestionar Favoritos.
**Actores**: Administrador, Ministro, Negocio, Embajador.
**Objetivo**: Permitir al usuario gestionar su lista de favoritos.
**Precondiciones**: El sistema debe estar disponible y haber ejecutado el caso de uso nº 6 por lo menos una vez.
**Post-condiciones**: El sistema debe reflejar las decisiones de gestión del usuario.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El usuario se dirige a la sección "favoritos" de su perfil. | 2. El sistema despliega la lista de favoritos del usuario agregando botones útiles al costado de cada uno. |
| 3. El usuario puede eliminar un favorito de su lista por medio de un botón o solamente desactivar las notificaciones. | |

Cursos Alternos:
- Ítem 2: El sistema no despliega ningún favorito al no haber agregado ninguno aun.

**8. Caso de uso**: Calificar Negocio.
**Actores**: Negocio, Embajador.
**Objetivo**: Permitir al usuario calificar un negocio a su parecer personal.
**Precondiciones**: El sistema debe estar y haber ejecutado el caso de uso nº 5.
**Post-condiciones**: El sistema guarda la calificación del usuario en el Negocio calificado.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El usuario decide calificar al negocio de 1 a 5 estrellas y agregar un comentario. | 2. El sistema guarda la calificación en la base de datos. |

Cursos Alternos:
- Ítem 1: Si la calificación no ha podido ser guardada, el usuario será notificado por una alerta.

#### 5.1.1.4 Negocio

El siguiente diagrama de casos de uso no incluye los casos de uso anteriormente expuestos pero cabe mencionar que son heredados por estar jerárquicamente más arriba que un visitante o embajador.

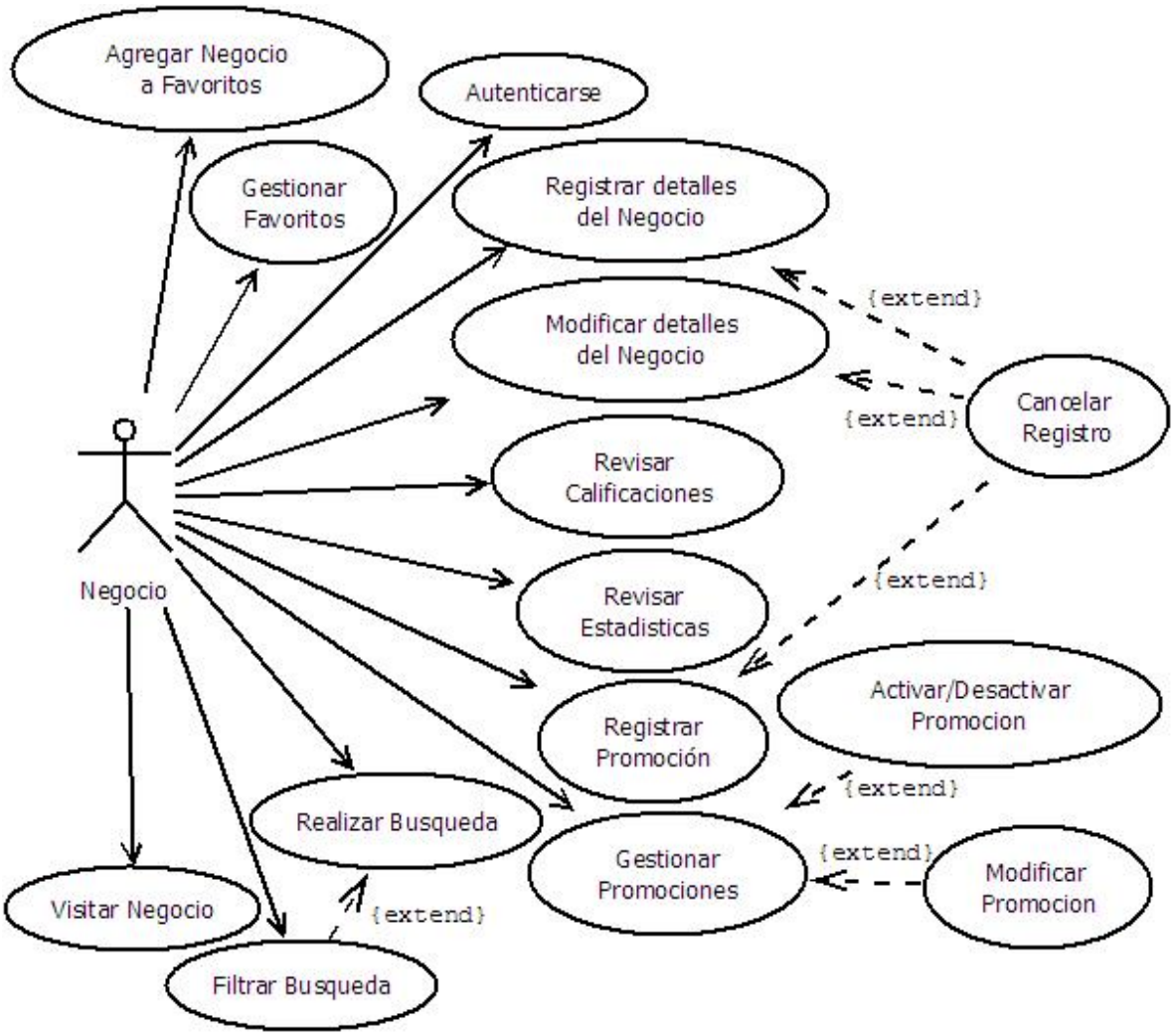



*Ilustración 11 Diagrama de Casos de Uso para un Negocio*

#### 5.1.1.5 Actores

- **Negocio:** Corresponde a las empresas o negocios pequeños adheridos a la campaña como Ministerios o Amigos.

#### 5.1.1.6 Especificación de los Casos de Uso

**9. Caso de uso**: Registrar detalles del Negocio.
**Actores**: Negocio.

**Objetivo**: Permitir al negocio registrar sus detalles.
**Precondiciones**: El sistema debe estar disponible.
**Post-condiciones**: El sistema guarda los detalles en la base de datos.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El Negocio ingresa los siguientes detalles en el formulario:<br><br>a. Nombre<br>b. Subtitulo<br>c. Descripción<br>d. Fotos<br>e. Descuento<br>f. Categoría<br>g. Dirección<br>h. País<br>i. Ciudad<br>j. Página Web<br>k. Horarios de Atención<br>l. Lista de Productos<br>m. Sucursales | 2. El sistema valida los datos ingresados. |
| | 3. El sistema sube los archivos seleccionados. |
| | 4. El sistema guarda los datos ingresados. |

Cursos Alternos:
- Ítem 2: Si algún dato es inválido será notificado por medio de una alerta.

**10. Caso de uso**: Modificar detalles del Negocio.
**Actores**: Negocio, Ministro.
**Objetivo**: Permitir al negocio modificar sus detalles.
**Precondiciones**: El sistema debe estar disponible y el usuario debe haber registrado sus detalles anteriormente.
**Post-condiciones**: El sistema guarda los detalles en la base de datos.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El Negocio modifica alguno de los siguientes detalles en el formulario:<br><br>a. Nombre<br>b. Subtitulo<br>c. Descripción<br>d. Fotos<br>e. Descuento<br>f. Categoría<br>g. Dirección<br>h. País<br>i. Ciudad<br>j. Página Web<br>k. Horarios de Atención<br>l. Lista de Productos<br>m. Sucursales | 2. El sistema valida los datos ingresados. |
| | 3. El sistema sube los archivos cambiados y elimina los quitados. |
| | 4. El sistema guarda los datos ingresados. |

Cursos Alternos:

- Ítem 2: Si algún dato es inválido será notificado por medio de una alerta.

**11. Caso de uso**: Registrar Promoción.
**Actores**: Negocio, Ministro.
**Objetivo**: Permitir al negocio registrar una nueva promoción.
**Precondiciones**: El sistema debe estar disponible y el usuario debe haber registrado sus detalles anteriormente.
**Post-condiciones**: El sistema guarda los detalles de la promoción en la base de datos.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El Negocio ingresa los detalles de la nueva promoción:<br><br>a. Titulo<br>b. Descripción<br>c. Fecha de termino<br>d. Descuento | 2. El sistema valida los datos ingresados. |
| | 3. El sistema guarda los datos ingresados en la base de datos. |
| | 4. El sistema alerta al usuario del éxito de la operación. |

Cursos Alternos:

- Ítem 2: Si algún dato es inválido será notificado por medio de una alerta.

**12. Caso de uso**: Gestionar Promociones.
**Actores**: Negocio, Ministro.
**Objetivo**: Permitir al negocio desactivar/activar promociones o modificar promociones.
**Precondiciones**: El sistema debe estar disponible y el usuario debe haber registrado una promoción.
**Post-condiciones**: El sistema guarda los cambios en la base de datos.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El Negocio puede modificar los detalles de las promociones y activar/desactivar promociones.<br>Al activar una promoción el usuario debe ingresar una nueva fecha de término. | 2. El sistema valida los datos ingresados. |
| | 3. El sistema guarda los datos ingresados en la base de datos. |
| | 4. El sistema alerta al usuario del éxito de la operación. |

Cursos Alternos:

- Ítem 2: Si algún dato es inválido será notificado por medio de una alerta.

**13. Caso de uso**: Revisar Calificaciones.
**Actores**: Negocio.
**Objetivo**: Permitir al negocio revisar las calificaciones recibidas, así como los comentarios adjuntos.
**Precondiciones**: El sistema debe estar disponible.
**Post-condiciones**: El sistema debe mostrar las calificaciones y comentarios hechos al negocio.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El Usuario se dirige a la sección calificaciones de si perfil. | 2. El sistema despliega una lista con todas las calificaciones y comentarios recibidos. |

**14. Caso de uso**: Revisar Estadísticas.
**Actores**: Negocio.
**Objetivo**: Permitir al negocio revisar sus estadísticas.
**Precondiciones**: El sistema debe estar disponible.
**Post-condiciones**: El sistema debe mostrar las calificaciones para el negocio.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El Usuario se dirige a la sección estadísticas de si perfil. | 2. El sistema muestra las estadísticas disponibles para el negocio.<br>a. N° Favoritos<br>b. N° de Visitas Mensuales |

#### 5.1.1.7 Ministro

Este actor hereda los casos de uso del Embajador y algunos de "Negocio" afectando solo a los los negocios que el mismo ha agregado al sistema. Es decir que los casos de uso "Modificar detalles de Negocio" y "Gestionar promociones" pueden ser usadas por los Ministros para moderar los contenidos publicados.

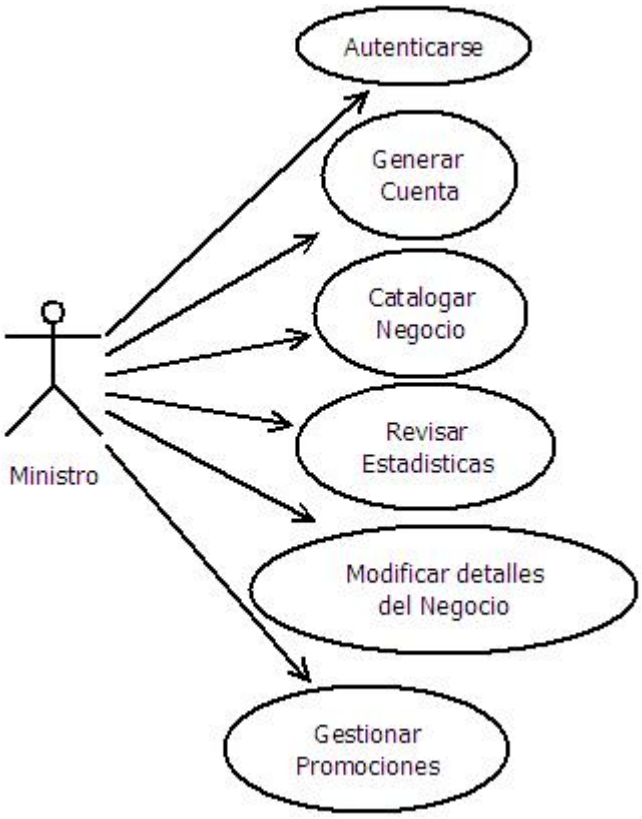



*Ilustración 12 Diagramas de Casos de Uso para un Ministro*

#### 5.1.1.8 Actores

- **Ministro:** Corresponde a la persona que se dedica a promocionar la campaña y agregar empresas o negocios pequeños a la misma.

#### 5.1.1.9 Especificación de los Casos de Uso

**15. Caso de uso**: Catalogar Negocio.
**Actores**: Ministro, Administrador.
**Objetivo**: Permitir al usuario catalogar al negocio como "Ministerio" o "Amigo" de la campaña.
**Precondiciones**: El sistema debe estar disponible.
**Post-condiciones**: El sistema debe guardar los cambios en la base de datos.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El usuario se dirige a la sección Negocios de su "oficina". | 2. El sistema despliega una lista de los negocios agregados por el Ministros o todos los negocios del sistema para el Administrador. |
| 3. El usuario realiza una búsqueda sobre la lista y cataloga los negocios necesarios como "Ministerio" o "Amigo" de la campaña. | 4. El sistema guarda los datos en la base de datos. |

**16. Caso de uso**: Revisar Estadísticas.
**Actores**: Ministro.
**Objetivo**: Permitir al Ministro revisar sus estadísticas.
**Precondiciones**: El sistema debe estar disponible.
**Post-condiciones**: El sistema debe mostrar las estadísticas útiles para el Ministro.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El usuario se dirige a la sección Estadísticas de su "oficina". | 2. El sistema despliega las estadísticas disponibles para Ministros:<br>a. Actividad de Negocios<br>b. Mostrar Negocios que no hayan iniciado sesión en un mes o más.<br>c. Problemas a nivel de usabilidad. |

**17. Caso de uso**: Generar Cuenta.
**Actores**: Ministro, Administrador.
**Objetivo**: Permitir Ministro generar cuentas para Negocios o nuevos Ministros.
**Precondiciones**: El sistema debe estar disponible.
**Post-condiciones**: El sistema guardar la nueva cuenta generada por el usuario.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El usuario puede generar una nueva cuenta señalando:<br>a. Tipo: "Ministro" o "Negocio"<br>b. Email<br>c. Contraseña | 2. El sistema valida los datos ingresados |
| | 3. El sistema alerta del éxito de la operación. |

Cursos Alternos:

- Ítem 2: Si algún dato es inválido será notificado por medio de una alerta.

#### 5.1.1.1 Administrador

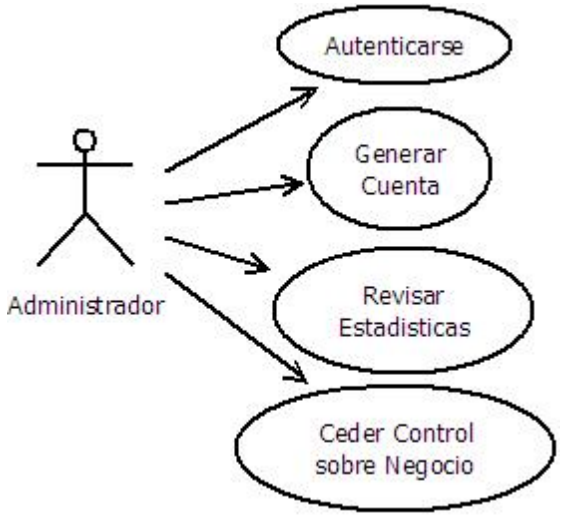



*Ilustración 13 Diagrama de Casos de Uso para un Administrador*

#### 5.1.1.2 Actores

- **Administrador:** Corresponde a la persona con la más alta jerarquía en la aplicación y usualmente será un técnico.

#### 5.1.1.3 Especificación de los Casos de Uso

**18. Caso de uso**: Revisar Estadísticas.
**Actores**: Administrador.
**Objetivo**: Permitir Administrador revisar sus estadísticas.
**Precondiciones**: El sistema debe estar disponible.
**Post-condiciones**: El sistema debe mostrar las estadísticas útiles para el Administrador.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El usuario se dirige a la sección Estadísticas de su "oficina". | 2. El sistema despliega las estadísticas disponibles para Administradores:<br>a. Embajadores Inactivos<br>b. Negocios inactivos.<br>c. Visitas Mensuales a la plataforma.<br>d. Problemas notificados por los clientes. |

**19. Caso de uso**: Ceder Control sobre Negocio.
**Actores**: Administrador.
**Objetivo**: Permitir Administrador quitar el control sobre un negocio y dárselo a otro Ministro.
**Precondiciones**: El sistema debe estar disponible.
**Post-condiciones**: El sistema debe guardar los cambios en la base de datos.

Escenario Principal:

| Acción del Actor | Respuesta del Sistema |
|---|---|
| 1. El usuario se dirige a la sección Ministros de su "oficina". | 2. El sistema despliega una lista de los Ministros registrados |
| 3. El usuario puede filtrar la lista para encontrar el Ministro deseado. | 4. El sistema refresca la lista mostrando una lista filtrada. |
| 5. El usuario selecciona el Ministro deseado y se dirige a la pestaña "Negocios bajo su control". | 6. El sistema muestra una lista con los Negocios bajo el control del Ministro. |
| 7. El usuario cliquea sobre "quitar control" y elige al Ministro que ahora tendrá el control sobre el Negocio y oprime Confirmar. | 8. El sistema guarda los cambios en la base de datos y confirma por medio de una alerta. |

## 5.2 Modelamiento de datos

Al utilizar una base de datos NoSQL, carente de un esquema fijo, se hace realmente difícil decidir de qué forma modelar los datos. Al analizar como MongoDB almacena los datos se ha decidido utilizar un diagrama de clases para mostrar de forma gráfica como se distribuyen los datos. Un diagrama de clases se hace lo más óptimo hasta el momento para esta tarea ya que MongoDB guarda "Objetos" dentro de sus Documentos. La única diferencia entre los Documentos y una clase como tal es la ausencia de métodos por parte de un Documento.

Al momento de estructuran datos en MongoDB y definir las relaciones entre estos se utilizan dos herramientas:

### 5.2.1 Referenciar

Las referencias almacenan las relaciones entre los datos al incluir vínculos entre documentos. Una técnica que puede compararse con el uso de claves primarias y secundarias en un sistema de bases de datos relacionales. Esta técnica da como resultado modelos de datos normalizados.

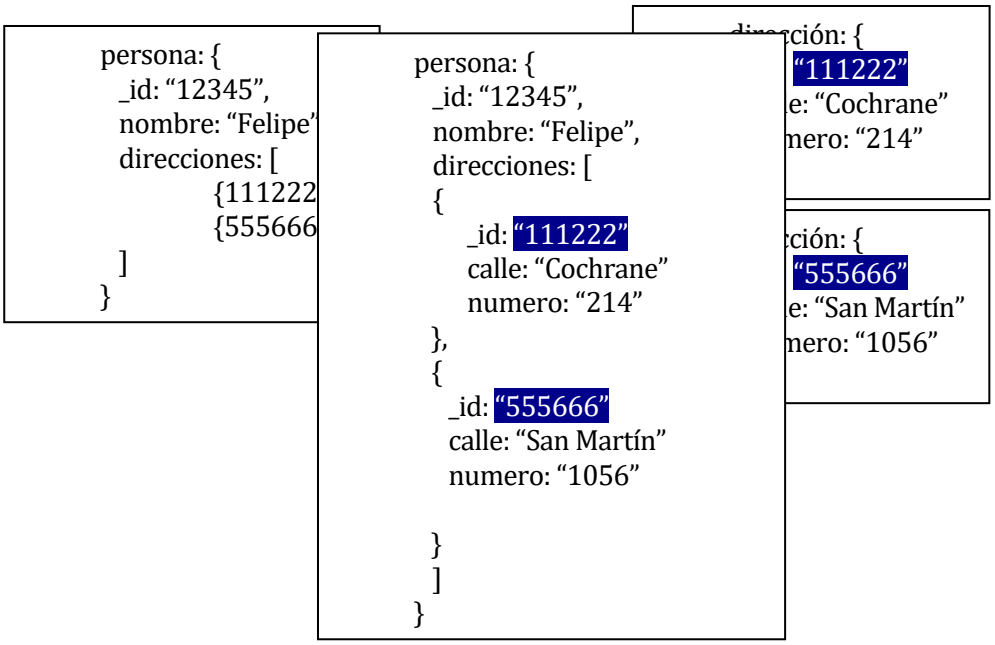



*Ilustración 14 Ejemplo de Referencias*

### 5.2.2 Embeber Datos

Al embeber documentos dentro de otros estamos capturando las relaciones al almacenar los datos dentro de un solo documento principal. Esta técnica la podemos comparar con la habilidad de cualquier lenguaje de programación orientado a objetos, donde un atributo puede ser otro objeto. Esta técnica da como resultado un modelo de datos des normalizado lo que permite a las aplicaciones obtener y manipular datos relacionados en una sola operación.

*Ilustración 15 Ejemplo de Embeber*

### 5.2.3 Traducción a UML

Entonces, conociendo estas técnicas podemos traducirlas a UML de la siguiente manera:

Documento = Clase
Referenciar = Asociación.
Embeber = Agregación.

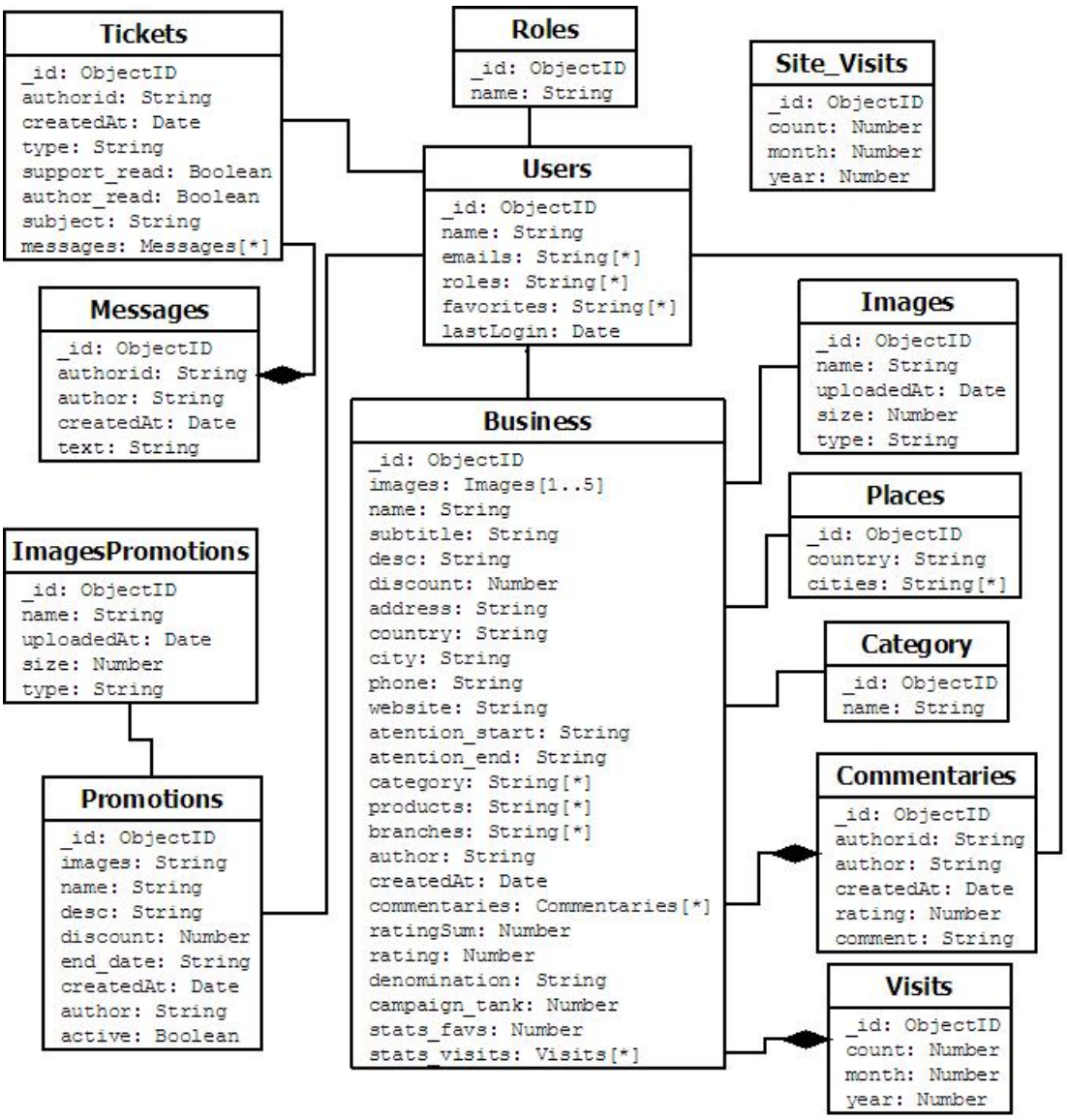



*Ilustración 16 Diagrama de Clases Genera*

# 6 DISEÑO

## 6.1 Diseño Físico de la Base de datos

A continuación, se mostrarán ejemplos de los documentos para cada colección en MongoDB y se explicaran los casos en que se utilizó la técnica de embeber o referenciar.

Cabe mencionar que algunas colecciones no fueron creadas ni diseñadas para este proyecto de software ya que son manejados directamente por Meteor por defecto. Estas colecciones son: Users, Roles, Images e ImagesPromotions. Sin embargo, al ser documentos flexibles pueden ser modificados en cualquier momento.

Se ha resuelto mostrar ejemplo de documentos en vez de modelos o esquemas dada la ausencia de restricciones en MongoDB por consiguiente un modelo UML no sería certero.

### 6.1.1 Users

Como ha sido mencionado anteriormente, esta colección es creada y manejada principalmente por el paquete *accounts-** de Meteor. Los atributos creados para este proyecto de software serán marcados con un asterisco (*).

```
{
 "_id": "gnNxeBij6gs3rBuK8",
 "services": {},
 "emails": [
  {
   "address": "embajador@prueba.cl"
  }
 ],
 "roles": [
  "normal-user"
 ],
 "status": {
  "online": true,
  "lastLogin": {
   "date": "2017-02-21T21:44:03.769Z",
   "ipAddr": "127.0.0.1",
   "userAgent": "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
                (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36"
  },
  "idle": false
 },
 "favorites": [
  {
   "id": "sPNAziwdZJDMMLxJ2",
   "name": "Govinda's Restaurant",
   "notifications": true
  }
 ]
}
```

*Ilustración 17 Documento Users*

#### 6.1.1.1 Atributos

- **_id:** Atributo que identifica cada documento.
- **services:** Documento embebido usado por el paquete *accounts-facebook* de Meteor para permitir la utilización de Facebook para registrarse y entrar al sistema.
- **emails:** Arreglo de documentos que guarda múltiples direcciones de correo y su estado de verificación. Este atributo es creado y manejado por el paquete *accounts-base* de Meteor.
- **roles:** Arreglo de cadenas de texto que almacena los roles proporcionados al usuario. Este atributo es creado y manejado por el paquete *roles* del usuario *alanning* de Meteor.
- **status:** Documento embebido que almacena el estado actual del usuario incluyendo la última fecha de inicio de sesión, aplicación usada para iniciar sesión y dirección ip. Este atributo es creado y manejado por el paquete *user-status* del usuario *mizzao* de Meteor.
- **favorites *:** Arreglo de documentos que almacena los negocios favoritos del usuario. Almacena una referencia al identificador del negocio, nombre y la opción de notificar al usuario sobre nuevas promociones. Atributo creado por el desarrollador de este proyecto de software. La des normalización de los datos es intencional de tal manera que al mostrar una lista de los favoritos en la aplicación no sean necesarias múltiples consultas por el nombre.

Existe otro atributo utilizado para almacenar datos públicos llamado **profile** que ha sido utilizado para almacenar el estado de registro para los usuarios con el rol de negocio. Es decir, si un negocio ha ingresado los datos de su negocio o no. Este atributo es manejado por el paquete *accounts-base* de Meteor pero no es utilizado para todos los usuarios y por la naturaleza de NoSQL estos atributos no son registrados en la base de datos si no existen al contrario de SQL donde dichos atributos toman el valor NULL.

### 6.1.2 Roles

Esta colección es creada y manejada por el paquete *roles* del usuario *alanning* de Meteor. Este ejemplo ha sido incluido en proyecto de software con el objeto de proporcionar información útil para comprender la implementación.

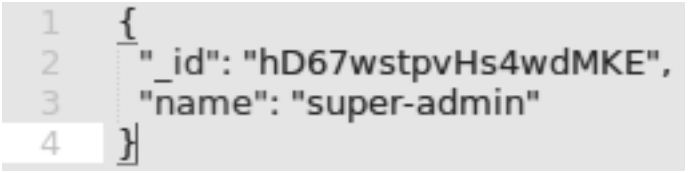



*Ilustración 18 Documento Roles*

#### 6.1.2.1 Atributos

- **_id:** Atributo que identifica cada documento.
- **name:** Atributo que almacena el nombre del rol.

### 6.1.3 Business

Esta colección almacena los datos de los negocios registrados en la plataforma.

```
{
  "_id": "sPNAziwdZJDMMLxJ2",
  "images": [
    "o6qyANTH8Amt94Ep7",
    "LJ5hRKZc4wyedTJvw"
  ],
  "name": "Govinda's Restaurant",
  "subtitle": "Vegetariano de Verdad",
  "desc": "Restaurant vegetariano y vegano donde podras encontrar
           almuerzos, especiales, pasteleria, panaderia, jugos naturales y
           kombucha.\nHare Krisna",
  "discount": 10,
  "address": "Cochrane 214 llegando a Salas",
  "country": "Chile",
  "city": "Concepción",
  "phone": "+56412795911",
  "website": "https://www.facebook.com/govindasveggieconcepcion/",
  "atention_start": "13:00",
  "atention_end": "18:00",
  "category": [
    "Restaurant",
    "Tiendas"
  ],
  "products": [
    "pan integral",
    "almuerzos",
    "hamburguesas",
    "queques",
    "galletones de avena",
  ],
  "author": "gJb5sAsiuChpkiKg2",
  "createdAt": "2017-02-13T14:18:00.877Z",
  "stats_visits": [
    {
      "count": 194,
      "month": 1,
      "year": 2017
    }
  ],
  "stats_favs": 2,
  "ratingSum": 5,
  "commentaries": [
    {
      "rating": 5,
      "comment": "Muy bueno, delicioso y barato",
      "authorId": "gJb5sAsiuChpkiKg2",
      "author": "negocio@prueba.cl",
      "createdAt": "2017-02-13T20:25:49.550Z"
    }
  ],
  "rating": 5,
  "denomination": "Ministerio",
  "campaign_rank": 5,
  "branches": [
    "Maipu 513, Concepción"
  ]
}
```


*Ilustración 19 Ejemplo Business*

#### 6.1.3.1 Atributos

- **_id:** Atributo que identifica cada documento.
- **images:** Arreglo de cadenas de texto las cuales almacenan referencias a los documentos de la colección "Images" como puede verse en la Ilustración 16.

- **name:** Atributo que almacena el nombre del negocio.
- **subtitle:** Atributo que da la posibilidad de almacenar un slogan o frase utilizada por el negocio.
- **desc:** Atributo que almacena una descripción del negocio.
- **discount:** Atributo que almacena el descuento ofrecido por el negocio.
- **address:** Atributo que almacena la dirección.
- **country:** Atributo que almacena el país donde reside el negocio.
- **city:** Atributo que almacena la ciudad donde está ubicado el negocio.
- **phone:** Atributo que almacena el número telefónico del negocio.
- **website:** Atributo que almacena la dirección web del negocio.
- **atention_start:** Atributo que almacena el horario de apertura del negocio.
- **atention_end:** Atributo que almacena el horario de cierre del negocio.
- **category:** Arreglo que almacena las varias categorías que corresponden al negocio.
- **products:** Arreglo que almacena los múltiples productos que comercializa el negocio.
- **author:** Atributo que almacena una referencia al identificador del usuario que ha creado el documento.
- **createdAt:** Atributo que almacena la fecha y hora de creación del documento.
- **stats_visits:** Arreglo de documentos embebidos que almacenan las visitas mensuales del negocio. Como puede observarse, la Clase Visits del diagrama de clases en la Ilustración 16 ha sido incluida como un sub documento.
- **stats_favs:** Atributo que almacena la cantidad de usuarios que tienen al negocio entre sus favoritos.
- **ratingSum:** Atributo que almacena la sumatoria de puntuaciones otorgadas por los usuarios en los comentarios.
- **commentaries:** Arreglo que almacena los comentarios realizados por los usuarios, así como las puntuaciones otorgadas. También se incluyen el identificador del usuario y nombre des normalizado. Puede observarse que al igual que el atributo *stats_visits* la clase Commentaries del diagrama de clases ha sido añadida al documento como un sub documento.
- **rating:** Atributo que almacena la puntuación promedio de las puntuaciones otorgadas en los comentarios.
- **denomination:** Atributo que almacena la denominación del negocio para la campaña OKI. Esta denominación puede ser “Amigo” o “Ministerio”.
- **campaign_rank:** Atributo que almacena el rango para la campaña otorgado a al negocio.
- **branches:** Arreglo que almacena las direcciones de las sucursales del negocio.

### 6.1.4 Category

Esta colección es utilizada para almacenar las categorías permitidas en la plataforma.

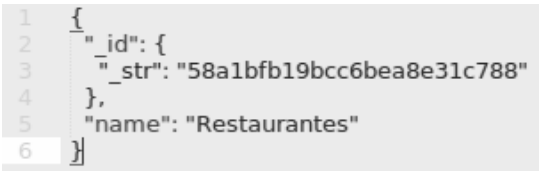



*Ilustración 20 Ejemplo Category*

#### 6.1.4.1 Atributos

- **_id:** Atributo que identifica cada documento.
- **name:** Atributo que almacena el nombre de la categoría.

### 6.1.5 Places

Esta colección es utilizada para guardar los países en donde funciona la campaña y sus ciudades.

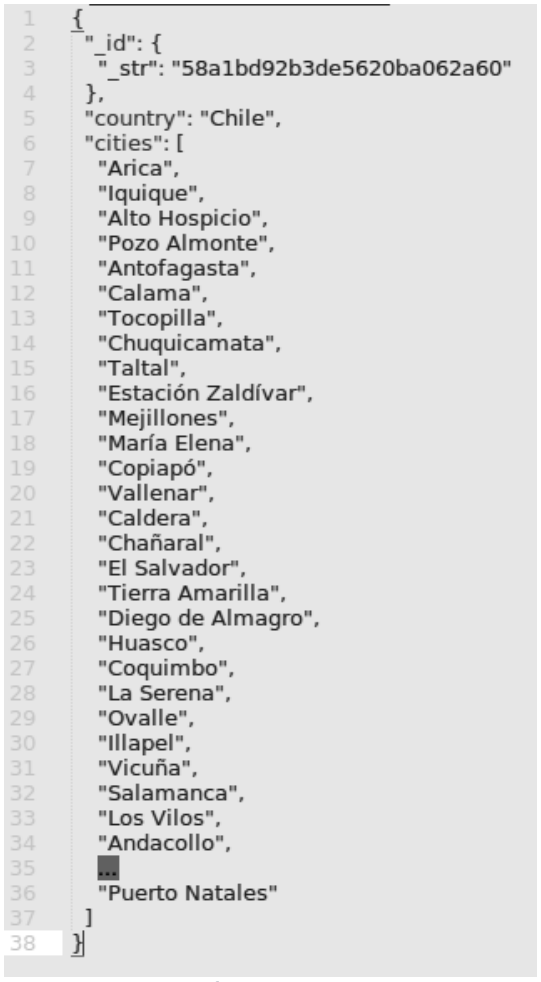



*Ilustración 21 Ejemplo Places*

#### 6.1.5.1 Atributos

- **_id:** Atributo que identifica cada documento.
- **country:** Atributo que almacena el nombre del país.

- **cities:** Arreglo que almacena las ciudades del país.

### 6.1.6 Promotions

En esta colección se almacenan las promociones creadas por los usuarios de tipo negocio.


```
{
  "_id": "zsdLg7DNqHn8jtfLA",
  "images": "cSivRryp26ezq9NdH",
  "name": "Fiesta de Empanadas",
  "desc": "Esta semana las empanadas son más ricas",
  "discount": 20,
  "end_date": "2017-02-17",
  "active": true,
  "createdAt": "2017-02-13T15:31:21.659Z",
  "author": "gJb5sAsiuChpkiKg2"
}
```


*Ilustración 22 Ejemplo Promotions*

#### 6.1.6.1 Atributos

- **_id:** Atributo que identifica cada documento.
- **images:** Arreglo de cadenas de texto las cuales almacenan referencias a los documentos de la colección "ImagesPromotions" como puede verse en la Ilustración 16.
- **name:** Atributo que almacena el nombre dado a la promoción.
- **desc:** Atributo que almacena una descripción de la promoción.
- **discount:** Atributo que almacena el descuento proporcionado durante la promoción.
- **end_date:** Atributo que almacena la fecha de término de la promoción.
- **active:** Atributo que almacena el estado actual de la promoción.
- **createdAt:** Atributo que almacena la fecha y hora de creación de la promoción.
- **author:** Atributo que almacena una referencia al identificador del usuario creador.

### 6.1.7 Images

Las imágenes son almacenadas en dos colecciones para separarlas entre imágenes de negocios y promociones, pero tienen exactamente la misma estructura por lo que solo será presentada una vez.

Esta colección es creada y manejada por el paquete *CollectionFS* de Meteor que permite almacenar archivos de varias formas, como:

- Nube S3 (Amazon Simple Storage Service).
- Nube Dropbox.
- Sistema de Archivos del sistema operativo.
- GridFS (Almacena archivos que exceden el tamaño máximo de un documento BSON directamente en la base de datos).


```
{
  "_id": "o6qyANTH8Amt94Ep7",
  "original": {
    "name": "govindascover6.jpg",
    "updatedAt": "2016-08-24T16:09:04.000Z",
    "size": 198401,
    "type": "image/jpeg"
  },
  "owner": "gJb5sAsiuChpkiKg2",
  "metadata": {
    "owner": "gJb5sAsiuChpkiKg2"
  },
  "uploadedAt": "2017-02-13T14:00:50.392Z",
  "copies": {
    "images": {
      "name": "govindascover6.jpg",
      "type": "image/jpeg",
      "size": 198401,
      "key": "images-o6qyANTH8Amt94Ep7-govindascover6.jpg",
      "updatedAt": "2017-02-13T14:00:50.403Z",
      "createdAt": "2017-02-13T14:00:50.403Z"
    }
  }
}
```


*Ilustración 23 Ejemplo Images*

#### 6.1.7.1 Atributos

Los atributos de esta colección no son explicados en la documentación de este paquete ya que son manejados completamente por la API del mismo.

### 6.1.8 Tickets

Esta colección almacena las peticiones de soporte además guarda los mensajes entre el usuario y el soporte técnico.

```
{
  "_id": "W8LzfMKeKMLJvWwEo",
  "type": "bugs",
  "subject": "Ticket de prueba",
  "messages": [
    {
      "text": "Mensaje de prueba",
      "authorId": "gJb5sAsiuChpkiKg2",
      "author": "negocio@prueba.cl",
      "createdAt": "2017-02-21T21:02:59.650Z"
    },
    {
      "text": "Mensaje de respuesta",
      "authorId": "TwBfX5WbYY3849Wfe",
      "author": "Paragati Das",
      "createdAt": "2017-02-21T21:07:08.281Z"
    }
  ],
  "authorId": "gJb5sAsiuChpkiKg2",
  "createdAt": "2017-02-21T21:02:59.644Z",
  "support_read": true,
  "author_read": true
}
```

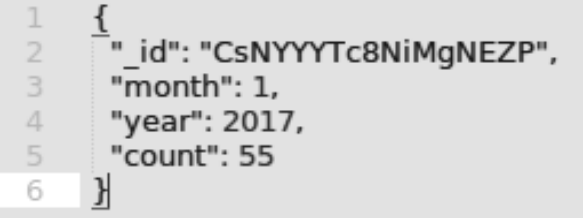
*Ilustración 24 Ejemplo Tickets*

#### 6.1.8.1 Atributos

- **_id:** Atributo que identifica cada documento.
- **type:** Atributo que almacena el tipo de problema presentado por el usuario.
- **subject:** Atributo que almacena el asunto de la petición de soporte.
- **messages:** Arreglo de sub documentos que almacena los mensajes entre el usuario y el soporte técnico. En este caso la clase Messages del diagrama de clases ha sido incluido como un sub documento en esta colección. Estos sub documentos incluyen:
    - **text:** Atributo que almacena el texto del mensaje.
    - **authorId:** Atributo que almacena una referencia al usuario que envía el mensaje.
    - **author:** Atributo que almacena el nombre o email del usuario que envía el mensaje.
    - **createdAt:** Atributo que almacena la fecha y hora de creación del mensaje.
- **authorId:** Atributo que almacena una referencia al identificador del usuario que crea la petición.
- **createdAt:** Atributo que almacena la fecha y hora de creación de la petición de soporte.
- **support_read:** Atributo que almacena si el soporte técnico ha leído la petición de soporte.
- **author_read:** Atributo que almacena si el autor ha leído la o las respuestas de la petición de soporte.

### 6.1.9 Visits

Esta colección almacena las estadísticas de las visitas a la plataforma.



```
{
  "_id": "CsNYYYTc8NiMgNEZP",
  "month": 1,
  "year": 2017,
  "count": 55
}
```


*Ilustración 25 Ejemplo Visits*

#### 6.1.9.1 Atributos

- **_id:** Atributo que identifica cada documento.
- **month:** Atributo que almacena el mes de la visita.
- **year:** Atributo que almacena el año de la visita.
- **count:** Atributo que almacena la cantidad de visitas a la plataforma.

## 6.2 Índices

Para mejorar la funcionalidad de búsqueda es necesaria la utilización de índices y con esto mantener una velocidad aceptable a lo largo de la mayor parte del ciclo de vida de la plataforma.

Esto se puede implementar de dos formas en nuestra plataforma. Una es desde la aplicación en sí y otra es desde la consola de MongoDB.

### 6.2.1 Desde Meteor

```
Meteor.startup(() => {
  Business._ensureIndex({
            "name" : "text",
            "products" : "text",
            "category" : "text"
  });
});
```

### 6.2.2 Desde la consola de MongoDB

```
db.business.ensureIndex({
            “name” : “text”,
            “products” : “text”,
            “category” : “text”
});
```

## 6.3 Diseño de interfaz

A continuación, se presentarán las maquetas de las pantallas más importantes de la plataforma, dando el esquema básico para el resto de la aplicación.

Se ha utilizado el freamework Bootstrap por lo que el estilo de los campos, botones, errores alertas, etc ya han sido pre diseñados por Twitter mejorando la velocidad de desarrollo.

### 6.3.1 Inicio de sesión y registro

Se ha presentado esta maqueta doble ya que la diferencia entre el inicio de sesión y el registro es solo de un campo que es la confirmación de la contraseña.

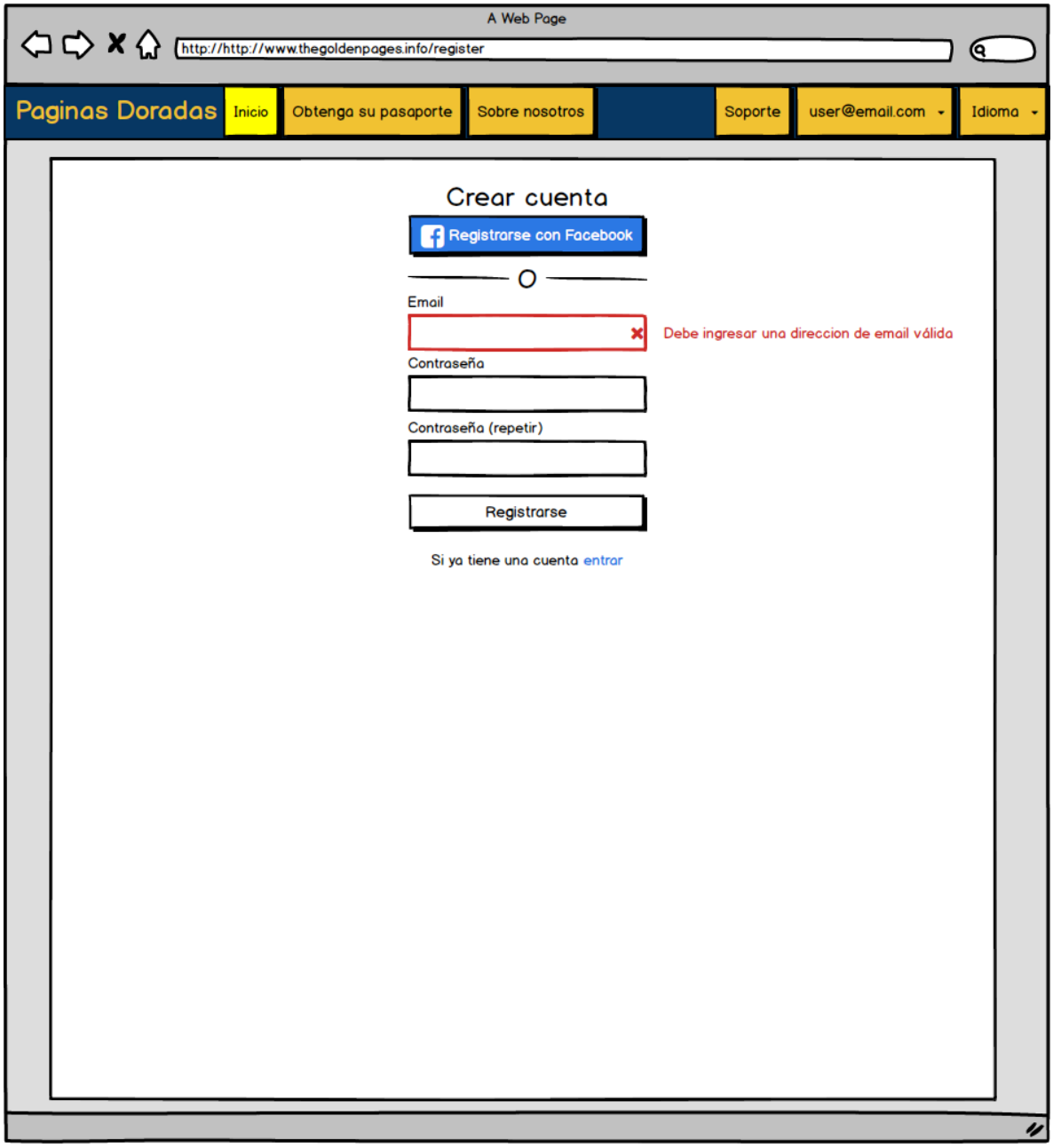
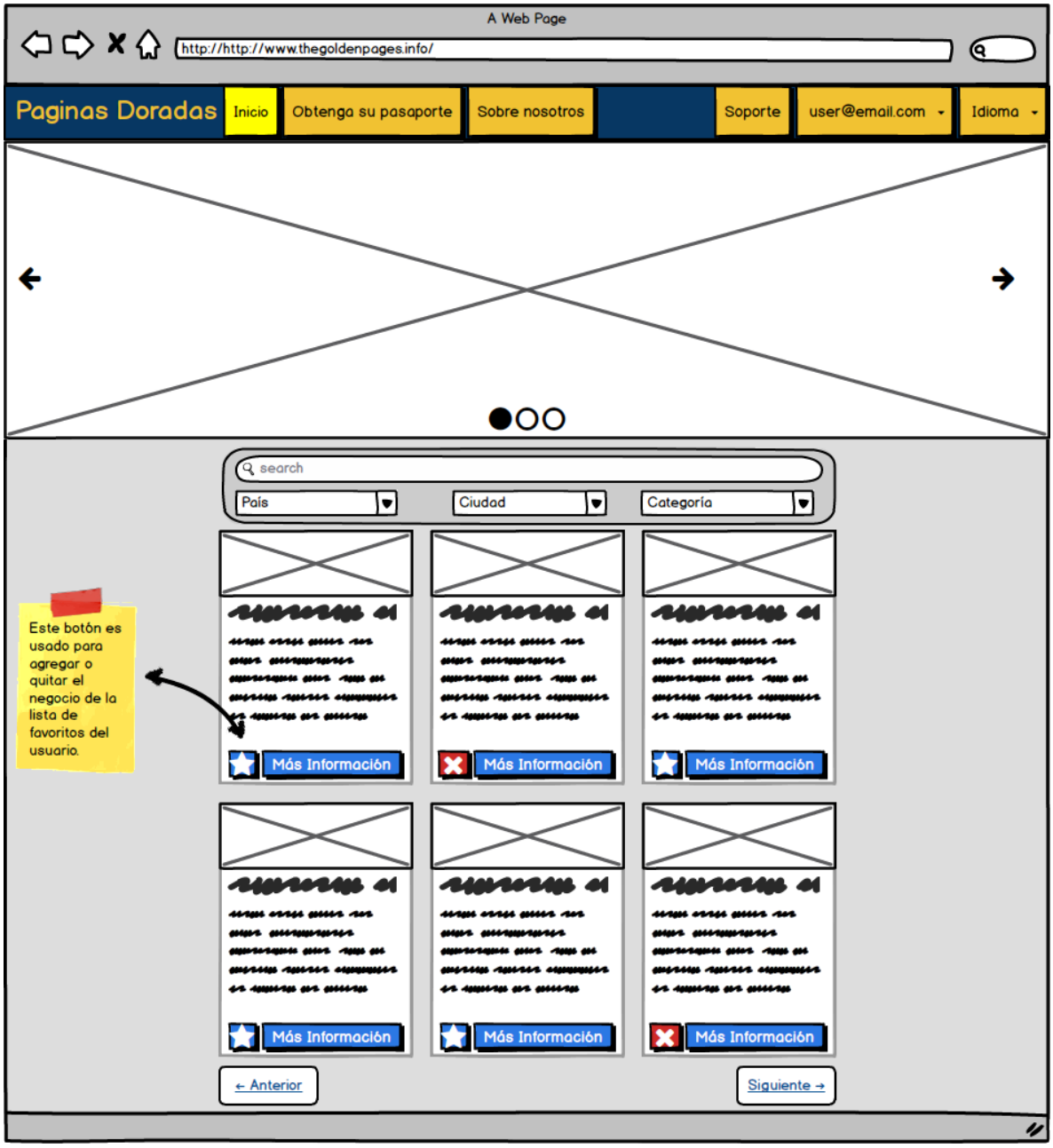



*Ilustración 26 Maqueta Inicio de Sesion y Registro*

### 6.3.2 Página principal

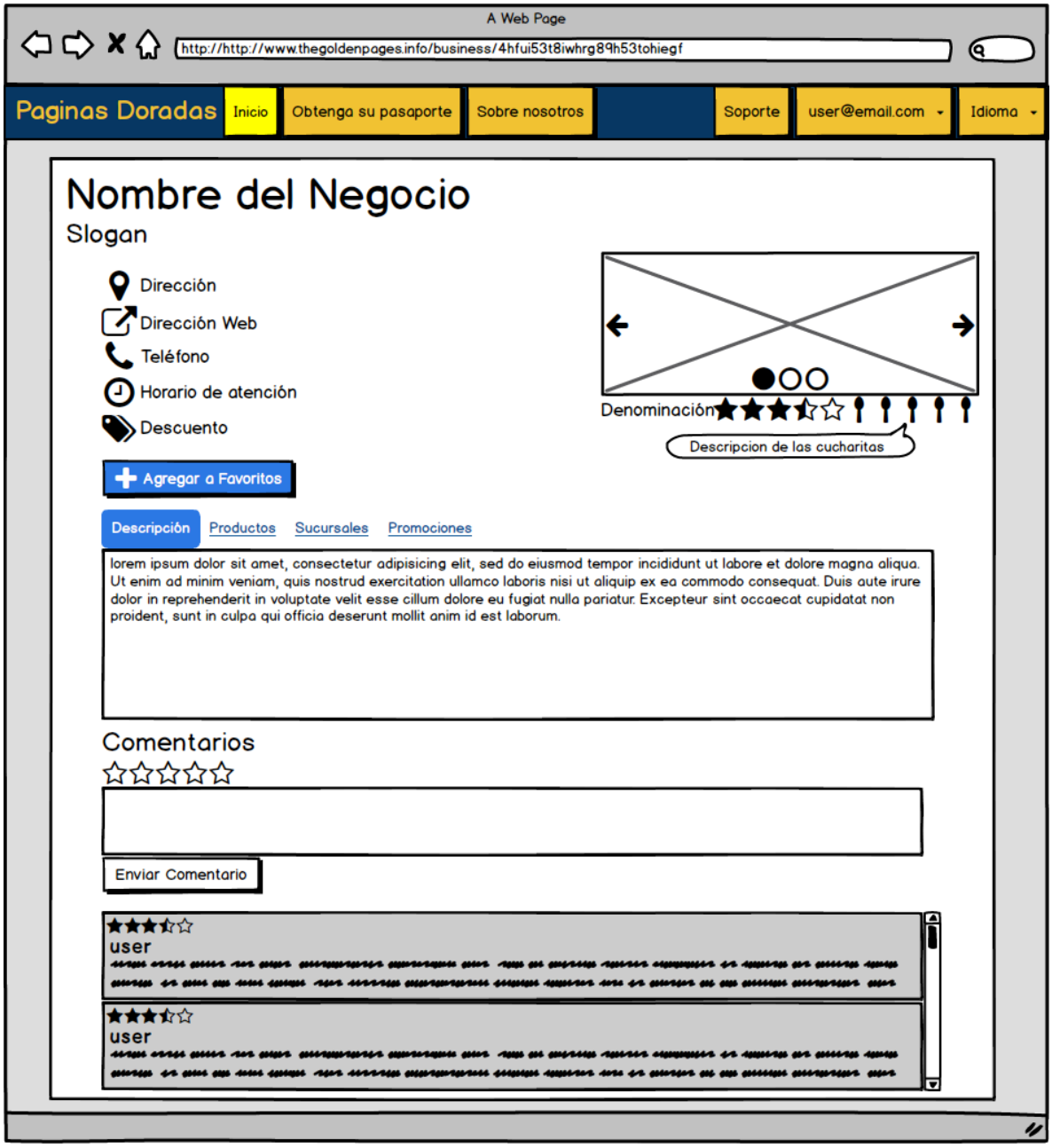
La barra de navegación superior es utilizada en toda la plataforma.
La sección superior al campo de búsqueda es un "slider" de fotografías y mensajes diversos.



### 6.3.3 Vista de un registro de negocio

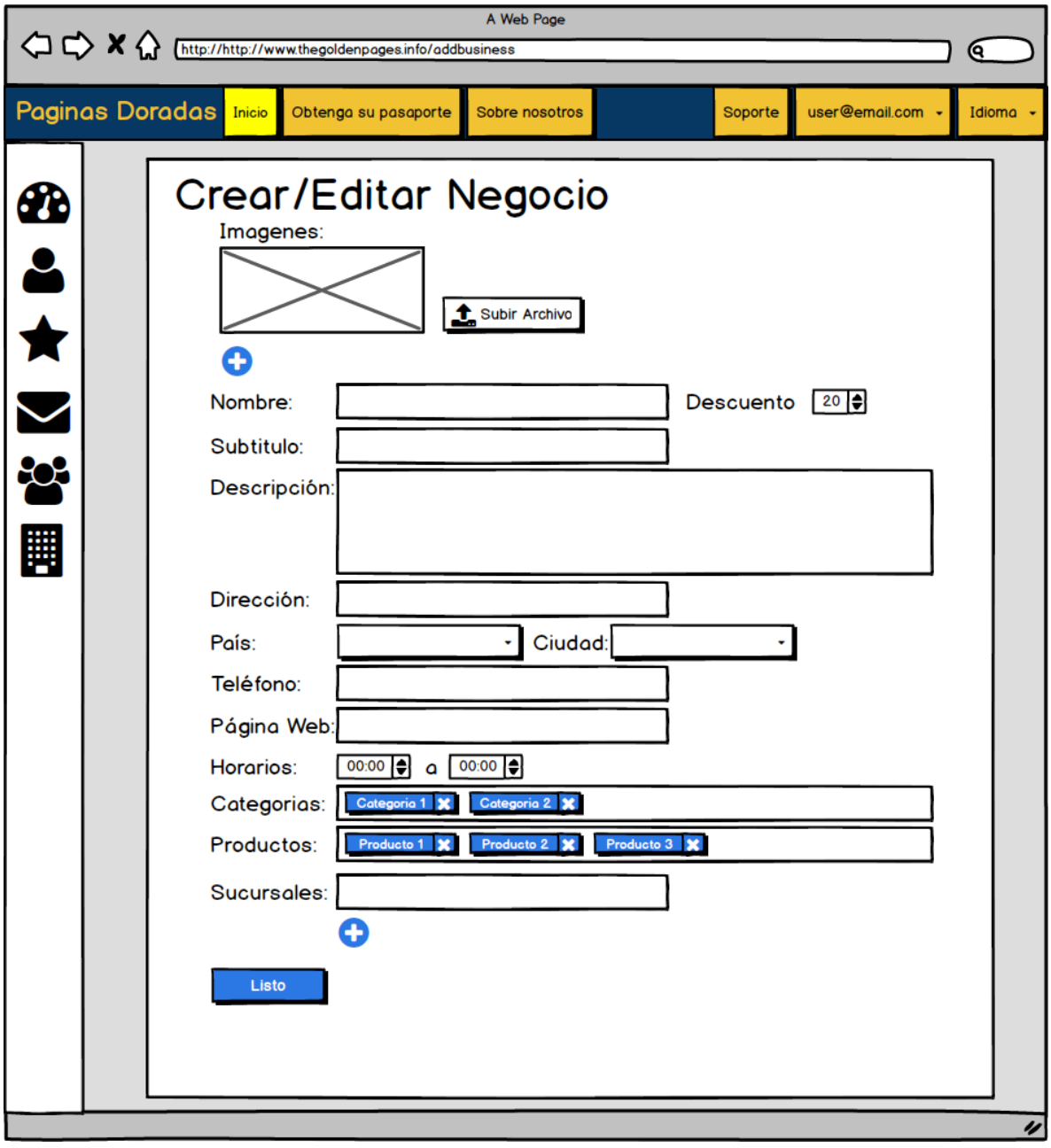
La burbuja bajo las cucharitas se trata de un “tooltip” informativo.

Las estrellas en la sección de comentarios son un selector de puntuación que el usuario quiera dar al negocio junto con un comentario.

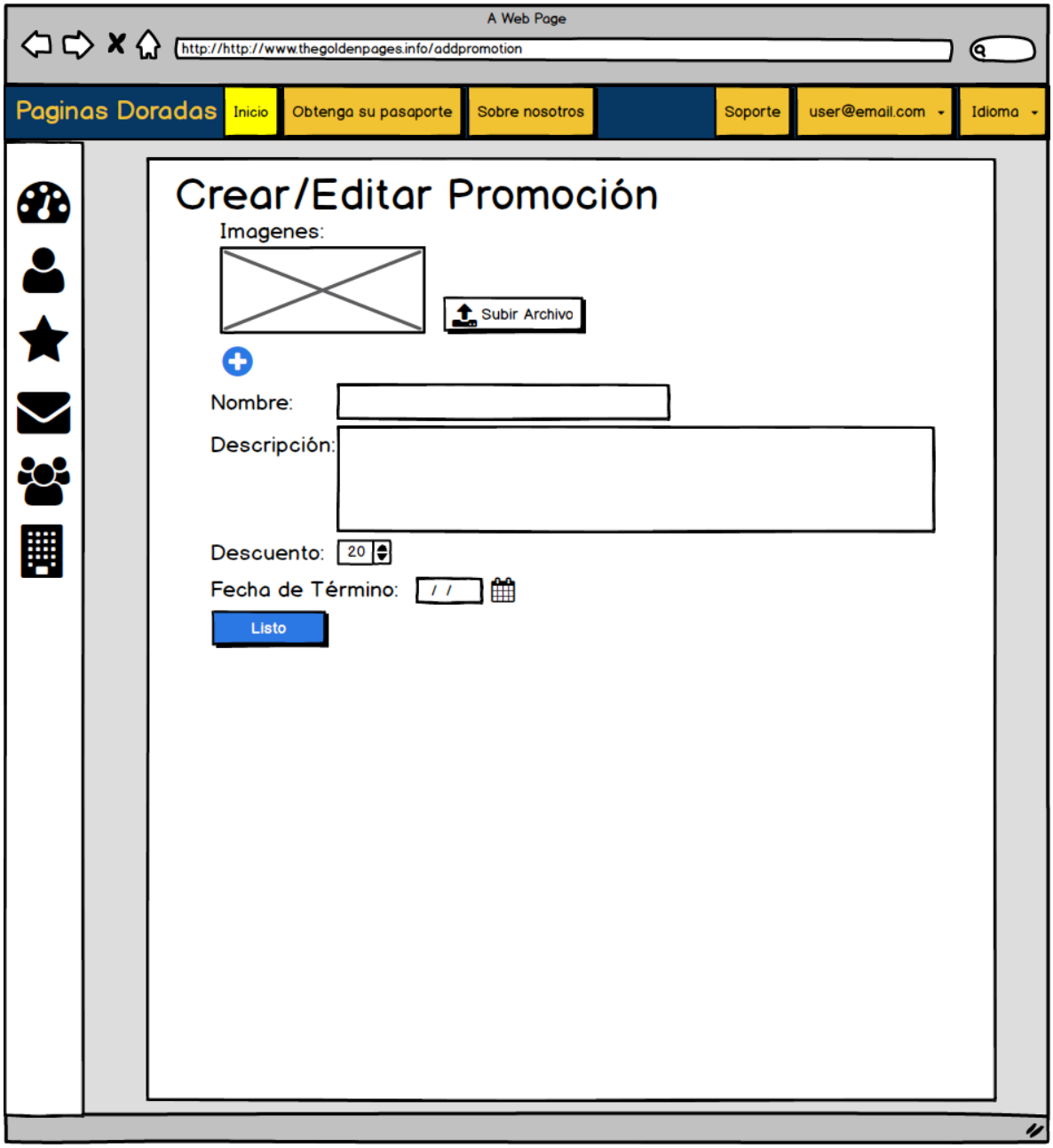


### 6.3.4 Crear o Editar un registro de negocio

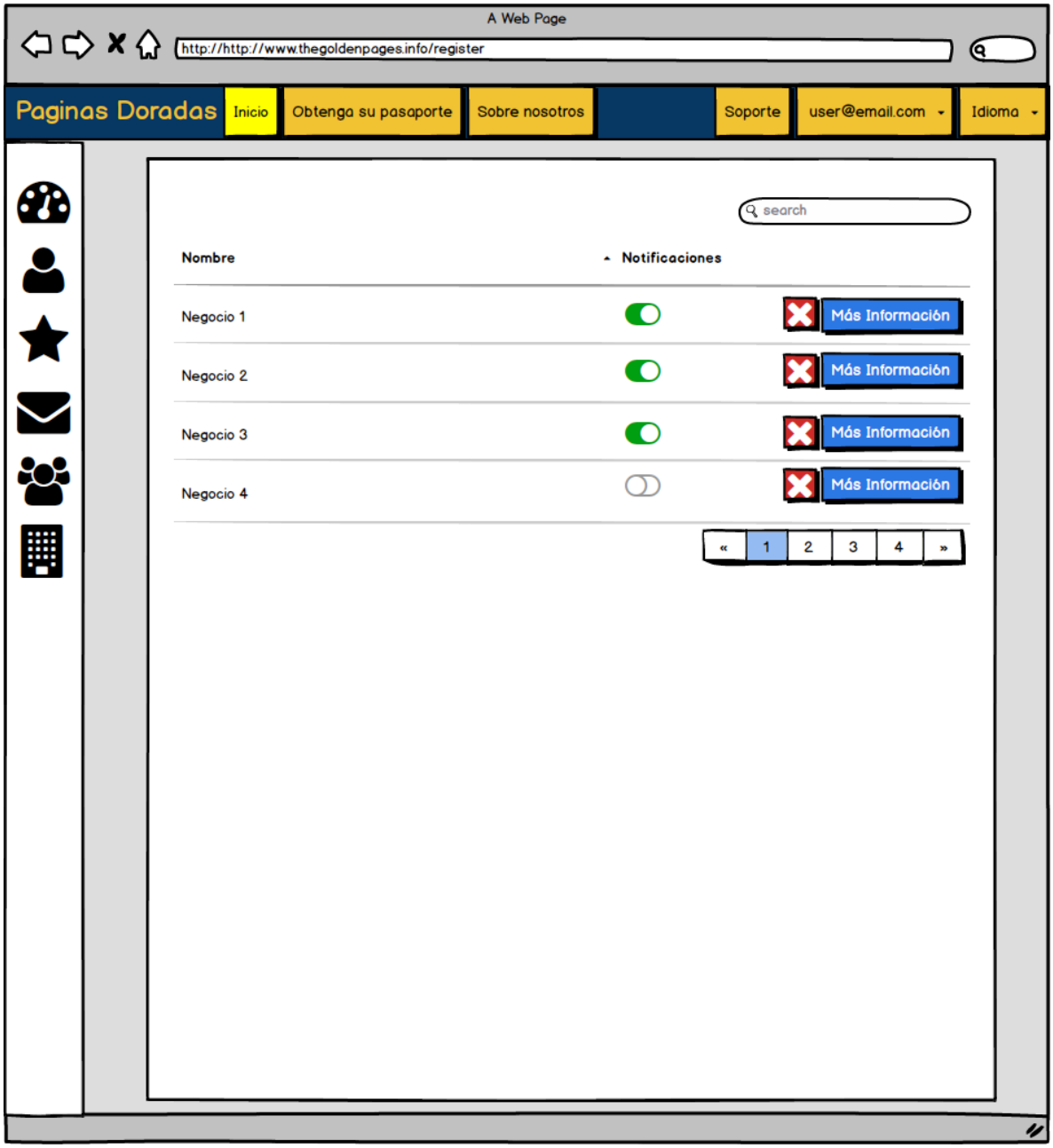
El botón con la cruz permite agregar más imágenes o sucursales según sea el caso.



### 6.3.5 Crear o Editar el registro de una promoción

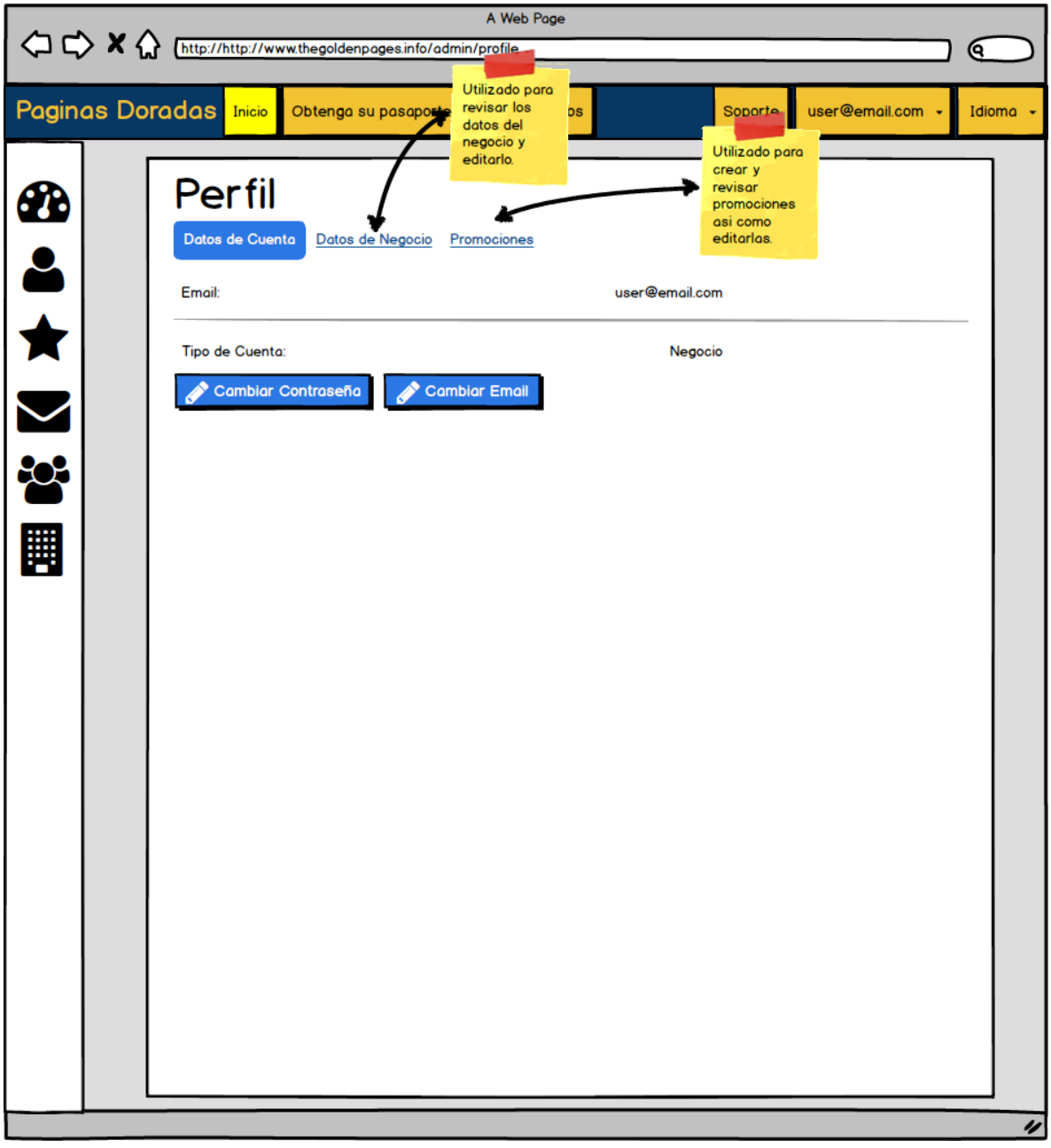


### 6.3.6 Administrar favoritos



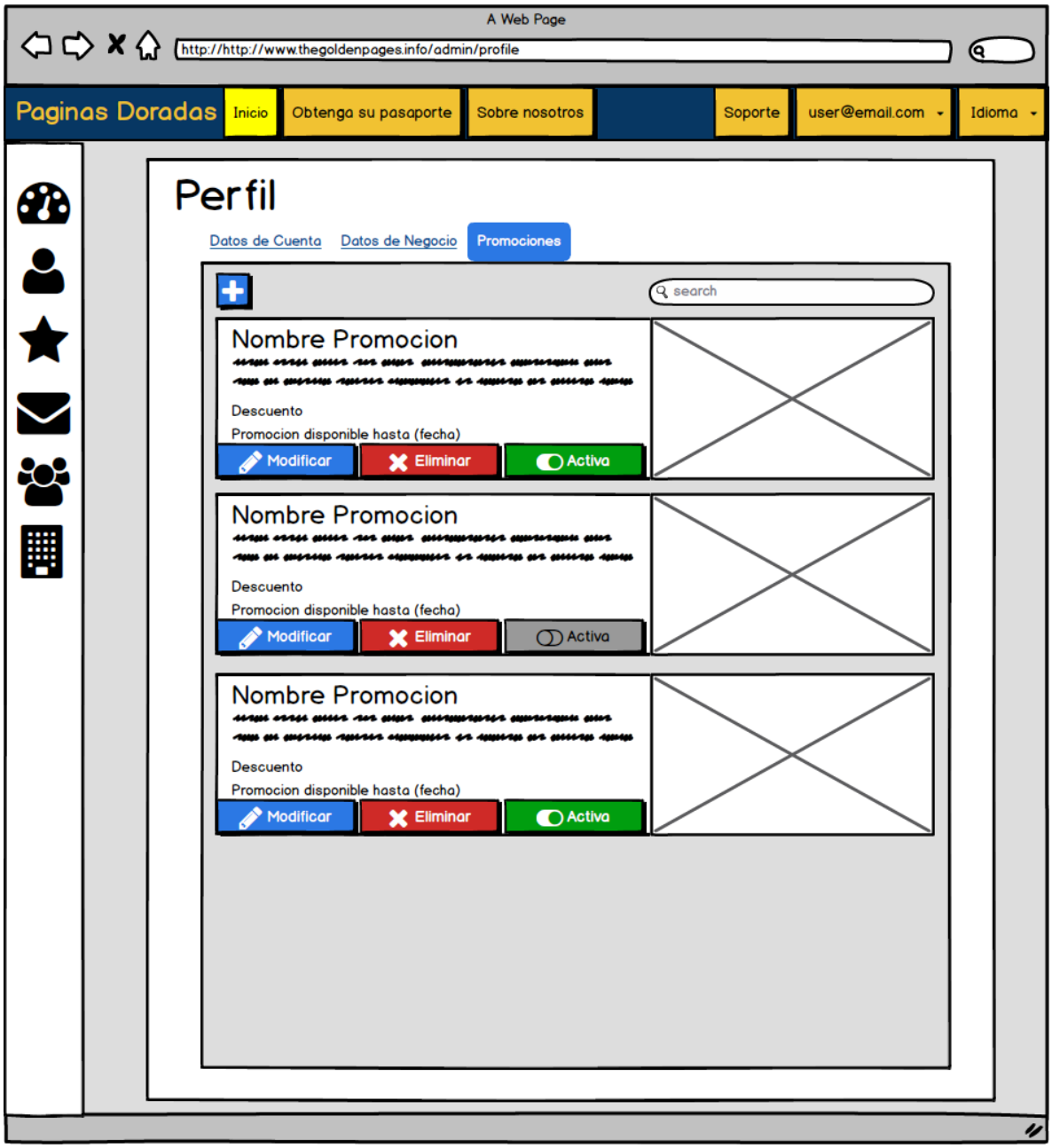
### 6.3.7 Perfil

Cabe aclarar que no todas las pestañas están presentes en todo momento sino en dependen del "Rol" del usuario. Es este caso se muestra el perfil para un usuario tipo negocio que muestra dos pestañas adicionales a los otros usuarios: Datos de Negocio y Promociones.



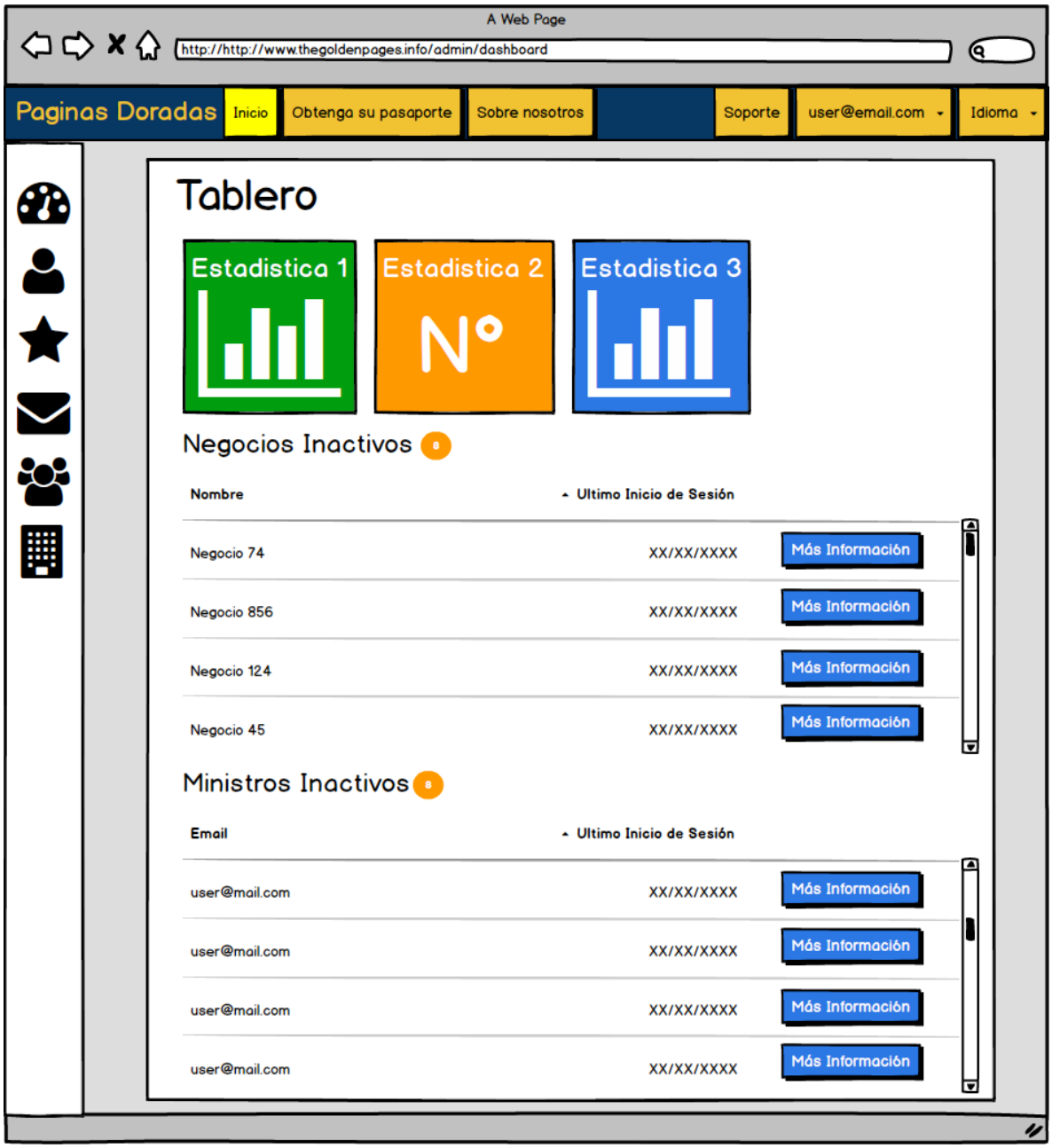
### 6.3.8 Perfil, Promociones

Maqueta específica para los usuarios de tipo negocio que permitirá crear, gestionar y revisar promociones.



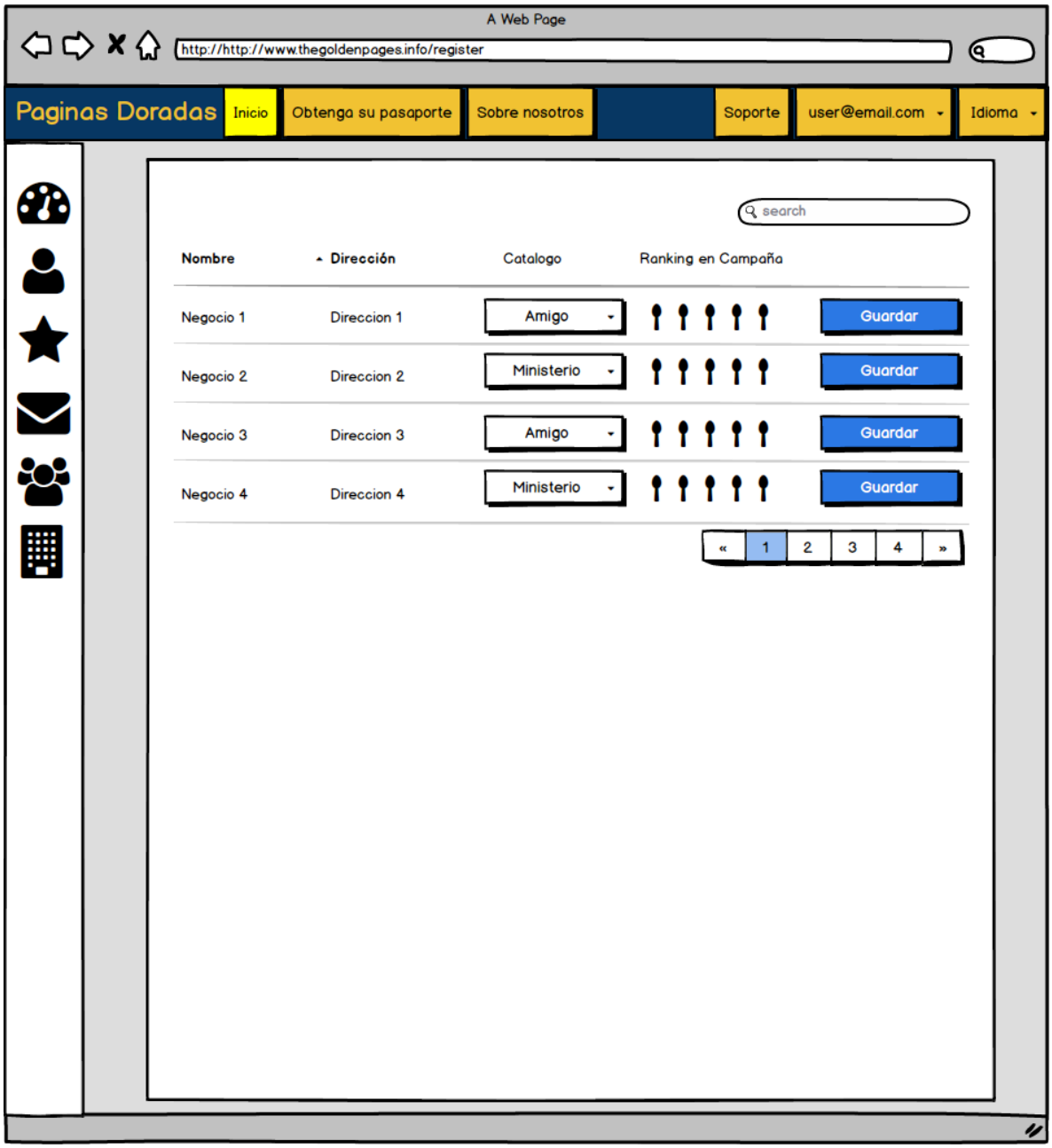
### 6.3.9 Tablero

Esta sección muestra información útil y estadísticas para los diferentes tipos de usuarios. Esta maqueta solo es un ejemplo de cómo se vería el tablero de un administrador general.



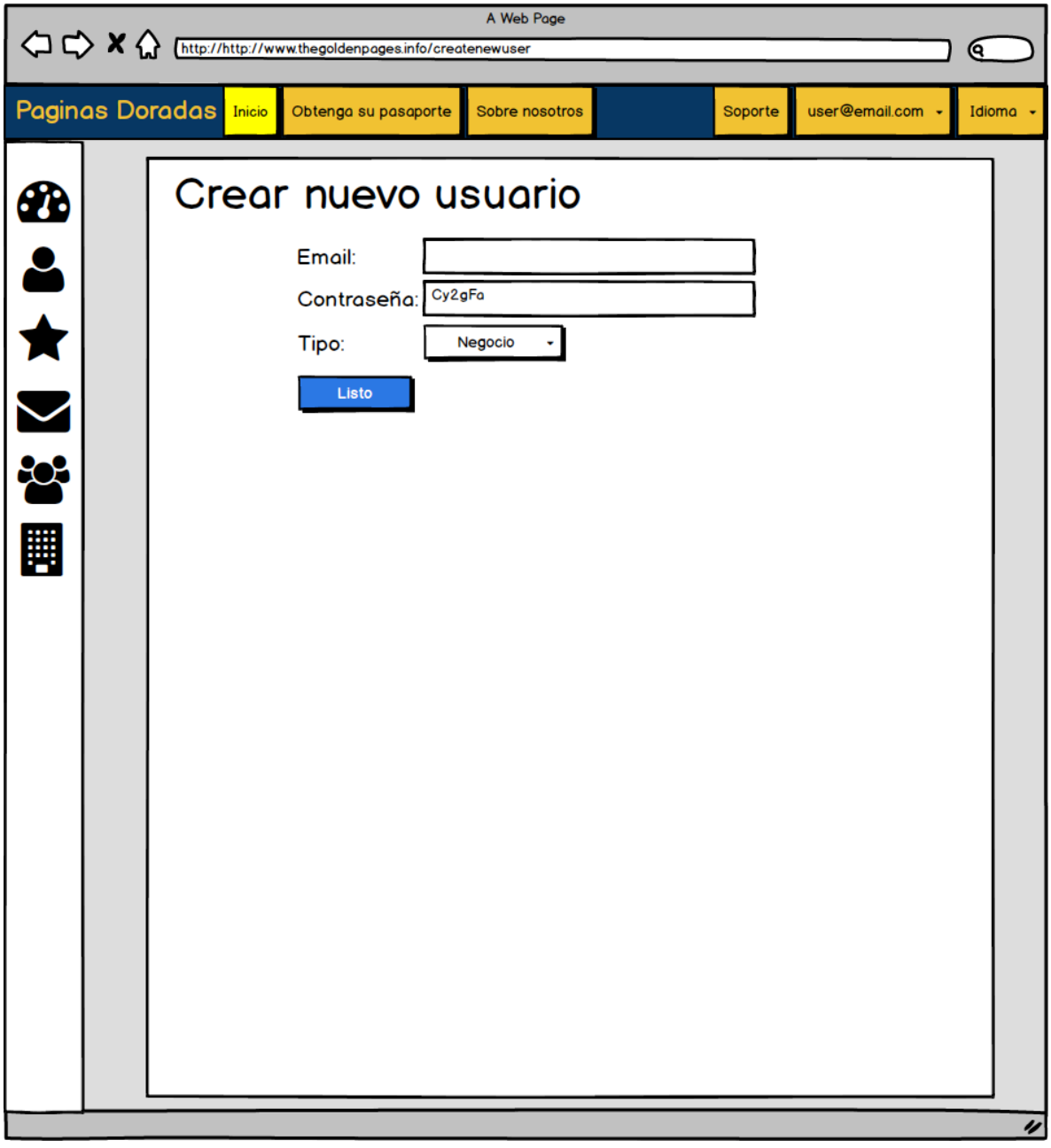
### 6.3.10 Catalogar Negocios

Esta sección permitirá catalogar los negocios como Amigos o Ministerios y proporcionarles una puntuación para la campaña dependiendo de su activismo.



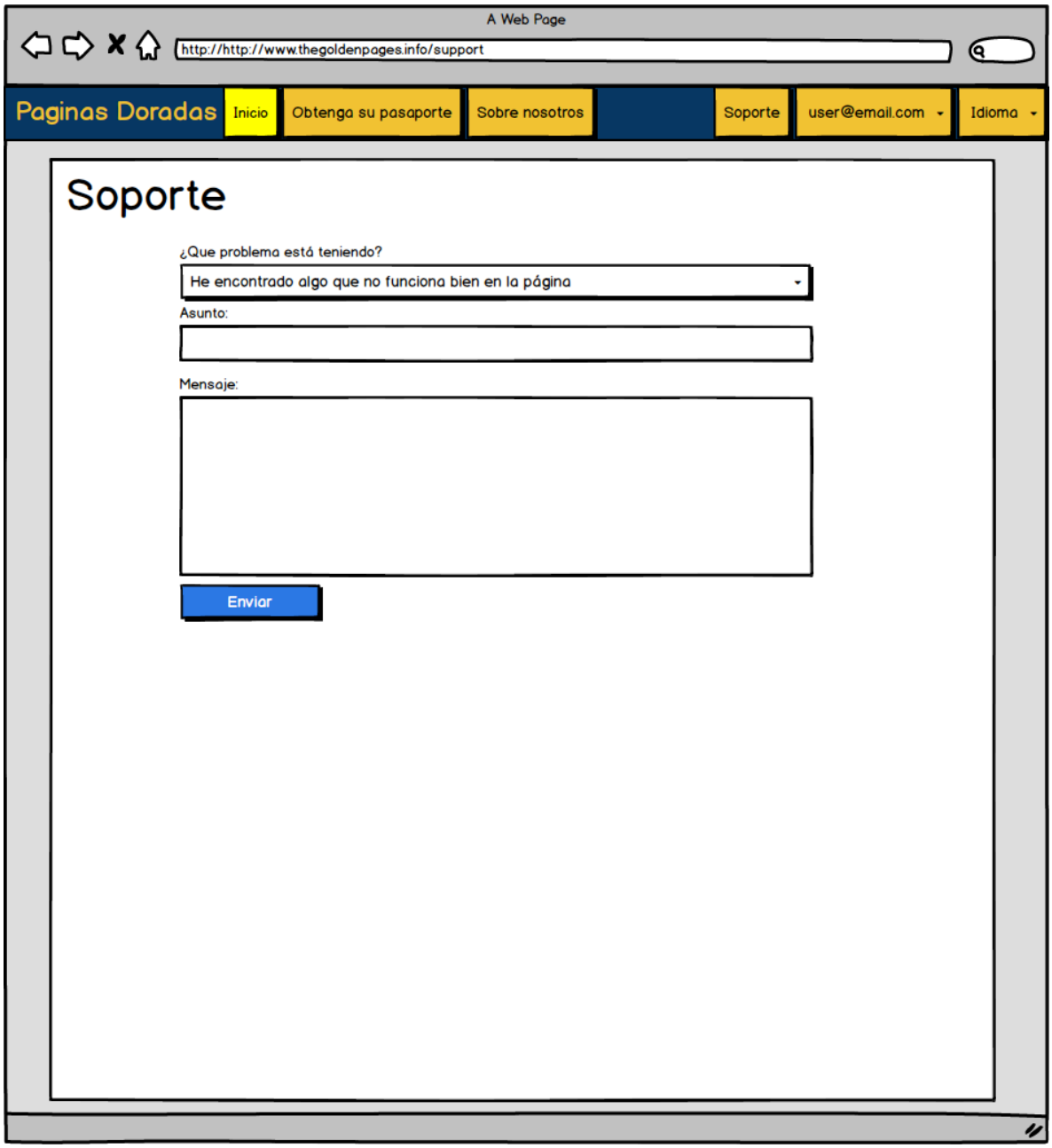
### 6.3.11 Crear nuevo usuario

Al contrario de los usuarios normales, los usuarios de tipo negocio o ministros deben ser creados en su sección aparte. Como se ve la contraseña debería ser autogenerada pero aun editable.



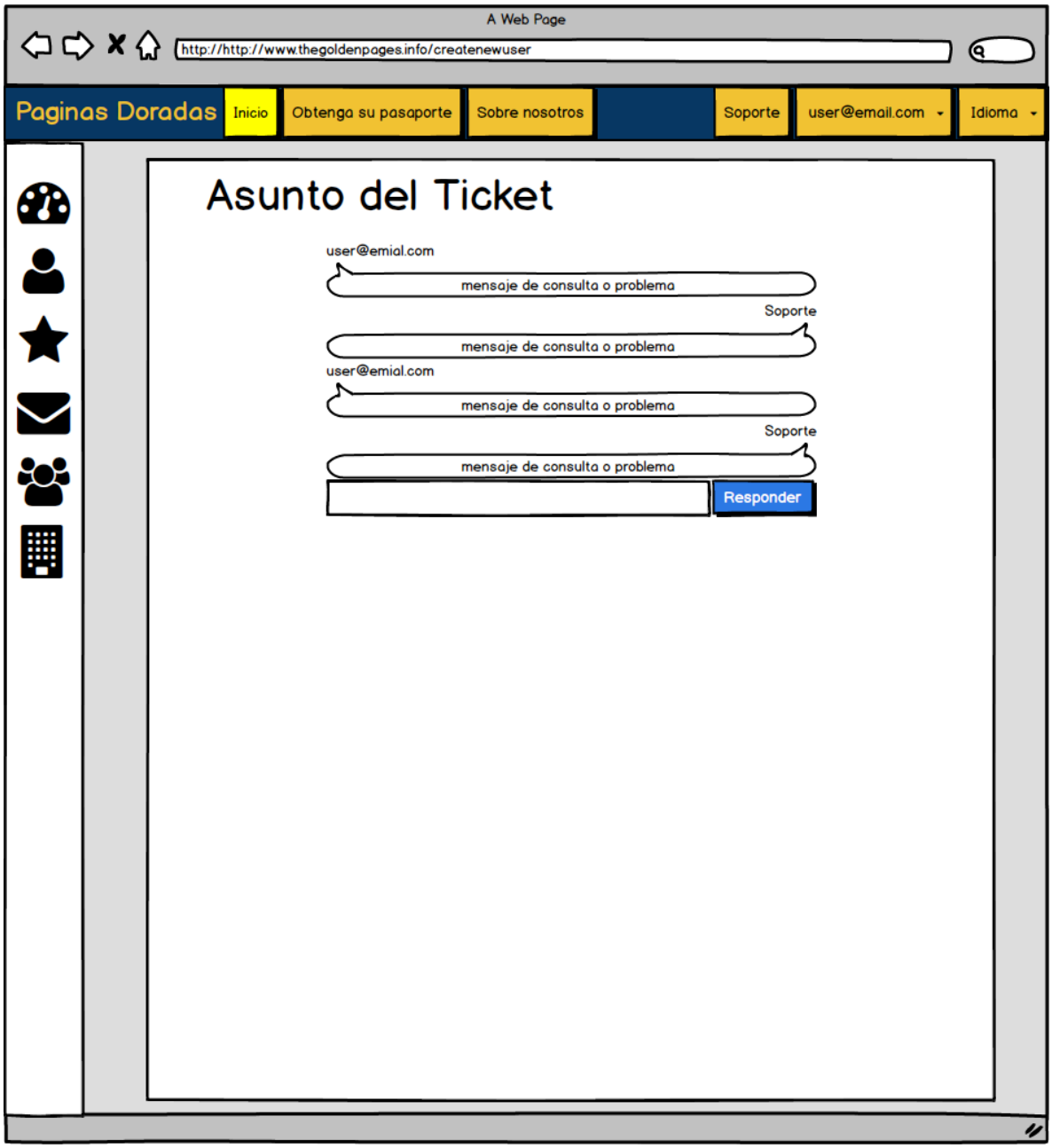
### 6.3.12 Enviar ticket de soporte

En esta sección el usuario puede enviar un ticket de soporte en caso de algún problema o duda de la plataforma.



### 6.3.13 Responder un ticket de soporte

Tanto el equipo de soporte como el mismo usuario deben ser capaces de responder el ticket como si de un chat se tratara para mejorar la posibilidad de solucionar el problema.



## 6.4 Código desarrollado

En esta sección se incluirá el código desarrollado para este proyecto de software.

### 6.4.1 Del lado del servidor

#### 6.4.1.1 Publish.js

```
Meteor.publish('allUsers', function() {
         return Meteor.users.find({}, {fields: {profile:1, 'emails.address':1,
roles:1, creator:1, favorites:1, status:1}});
})

Meteor.publish("search", function(searchValue, skipCount){
         if(!searchValue){
                  return Business.find({}, {limit: 6, skip: skipCount});
         }else{
                  return Business.find(
                  {
                           $text: {
                                    $search: searchValue,
                                    $language: "es",
                           }
                  },
                  {
                           fields: {
                                    score: { $meta: "textScore" }
                           },
                           sort: {
                                    score: { $meta: "textScore"}
                           },
                           limit: 6,
                           skip: skipCount
                  }
                  );
         }
});
```

#### 6.4.1.2 Métodos y configuraciones

Estos métodos son luego usados por el lado del cliente y son utilizados para mejorar la seguridad de la aplicación al ejecutarse del lado del servidor.

- **SiteVisitTracket.js**

```
Meteor.methods({
    trackSiteVisits: function(){
            date = new Date();
            Visits.upsert({
                        month: date.getMonth(),
                        year: date.getFullYear()
                },{
                        $set: {
                                month: date.getMonth(),
                                year: date.getFullYear()
                        },
                        $inc: {
                                count: 1
                        }
                });
    }
});
```

- **businessVisitorStats.js**

```
Meteor.methods({
    trackVisit: function(businessId){
            date = new Date();
            update = Business.update({ _id: businessId,
                    stats_visits: {
                            $elemMatch: {
                                    month: date.getMonth(),
                                    year: date.getFullYear()
                            }
                    }
            }, {
                    $set: {
                            "stats_visits.$.month" : date.getMonth(),
                            "stats_visits.$.year": date.getFullYear()
                    },
                    $inc: {
                            "stats_visits.$.count": 1
                    }
            });
            //En caso de que no se encuentre un candidato que coincida con
    el mes y año, o sea que no existe entonces...
            if(update == 0){
                    Business.update({_id : businessId}, {
                            $push: {
                                    stats_visits: {
                                            month: date.getMonth(),
                                            year: date.getFullYear(),
                                            count: 1
                                    }
                            }
                    });
            }
    }
});
```

- **changePassword.js**

```
Meteor.methods({
    changeEmail: function(userId, oldEmail, newEmail){
            Accounts.addEmail(userId, newEmail, false);
                Accounts.removeEmail(userId, oldEmail);
    }
});
```

- **createAccount.js**

```
Meteor.methods({
    createUsers: function(email, password, role) {
            if(email && password && role){
                var userId =
                        Accounts.createUser({
                                email: email,
                                password: password
                        });
                        Meteor.users.update({_id: userId}, {
                                $set: {
                                        creator:
Meteor.userId()
                                }
                        });
                        Roles.addUsersToRoles(userId, role);
            }
    },
    defaultRoleSet: function(){
            Roles.addUsersToRoles(this.userId, 'normal-user');

    }
 });

var postSingUp = function(userId) {
    Roles.addUsersToRoles(userId, 'normal-user');
};

AccountsTemplates.configure({
    postSignUpHook: postSingUp
});
```

- **favorites.js**

```
Meteor.methods({
    addFav: function(userId, businessId, businessName){
        Meteor.users.update(userId, {
            $push: {
                favorites: {
                  id: businessId,
                  name: businessName,
                  notifications: true
                }
            }
        });
        Business.update(businessId, {
            $inc: {
                stats_favs: 1
            }
        });
    },
```

```
        removeFav: function(userId, businessId){
            Meteor.users.update(userId, {
                $pull: {
                    favorites: {
                        id: businessId
                    }
                }
            });
            Business.update(businessId, {
                $inc: {
                    stats_favs: -1
                }
            });
        },

        favToggle: function(userId, favId, notificationState){
            Meteor.users.update({_id:userId, "favorites.id":favId}, {
                $set: {
                    "favorites.$.notifications": !notificationState
                }
            });
        }
    });
```

- **main.js**

```
import { Meteor } from 'meteor/meteor';
//JS
import '/imports/lib';
import '/imports/collections';
import './publish.js';

Images.allow({
  'insert': function () {
    // add custom authentication code here
    return true;
  }
});

Meteor.startup(() => {
    Business._ensureIndex({
             "name" : "text",
             "products" : "text",
             "category" : "text"
    });
});
```

#### 6.4.1.3 Colecciones

A continuación, se presentan las configuraciones y restricciones aplicadas a las colecciones de MongoDB en el lado de la aplicación.

- **Business.js**

```
Business = new Mongo.Collection("business");

Category = new Mongo.Collection("category");

Places = new Mongo.Collection("places");

Images = new FS.Collection("images", {
            stores: [new FS.Store.FileSystem("images", {path:
"~/golden_pages_uploads/images/business"})],
            filter: {
                        maxSize: 5242880,
                        allow: {
                                    contentTypes: ['image/*'],
                        },
                        onInvalid: function (message) {
                                    if (Meteor.isClient) {
                                                Bert.alert("El archivo elegido debe
ser una imagen", 'danger', 'growl-top-right');
                                    } else {
                                                console.log(message);
                                    }
                        }
            }
});

Images.allow({
    download: function(userId, fileObj) {
        return true;
    }
});

Business.allow({
    insert: function(userId, doc){
            return !!userId;
    },
    update: function(userId, doc){
            return !!userId;
    }
});

Commentaries = new SimpleSchema({
    authorId: {
            type: String,
            autoValue: function() {
                        return Meteor.userId();
            },
            autoform: {
                        type: "hidden",
            }
    },
    author: {
            type: String,
            autoValue: function() {
```

```
                        if(Meteor.user()){
                                if(Meteor.user().profile){
                                        if(Meteor.user().profile.name){
                                                return
    Meteor.user().profile.name;
                                        }else
                                                return
    Meteor.user().emails[0].address;
                                }
                                else{
                                        return
    Meteor.user().emails[0].address;
                                }
                        }
                },
                autoform: {
                        type: "hidden",
                }
        },
        createdAt: {
                type: Date,
                autoValue: function() {
                        return new Date()
                },
                autoform: {
                        type: "hidden",
                }
        },
        rating: {
                type: Number,
                autoform: {
                        type: "hidden"
                }
        },
        comment: {
                type: String,
                autoform: {
                        type: "hidden"
                }
        }
    });

    Visits = new SimpleSchema({
        count: {
                type: Number,
        },
        month: {
                type: Number,
        },
        year: {
                type: Number,
        }
    });

    var Schemas = {};

    Schemas.Business = new SimpleSchema({
        images: {
                type: [String],
                minCount: 1,
                maxCount: 5,
```

```
                label: "Imagenes"
        },
        "images.$": {
                autoform: {
                        afFieldInput: {
                                type: "fileUpload",
                                collection: "images",
                                label: "Elija una imagen que represente su
negocio" ,
                                accept: 'image/*'
                        },
                        placeholder: "Elija una imagen que represente su
negocio"
                }
        },
        name: {
                type: String,
                label: "Nombre",
                max: 30,
                autoform: {
                        placeholder: "Ej: Tienda Naturista"
                }
        },
        subtitle: {
                type: String,
                label: "Subtitulo",
                max: 30,
                optional: true,
                autoform: {
                        placeholder: "Ej: Slogan"
                }
        },
        desc: {
                type: String,
                label: "Descripcion",
                max: 140,
                autoform: {
                        afFieldInput: {
                                type: "textarea"
                                },
                        placeholder: "Esta descripcion tiene un maximo de 140
caracteres"
                }
        },
        discount: {
                type: Number,
                label: "Descuento",
                min: 0,
                max: 100,
                autoform: {
                        placeholder: "Descuento ofrecido a nuestros usuarios.
Entre 0% a 100%"
                }
        },
        address: {
                type: String,
                label: "Direccion",
                max: 50,
                autoform: {
                        placeholder: "Ej: Calle, Nº 999"
                }
```

```
        },
        country: {
                type: String,
                label: "Pais",
                autoform: {
                        type: "universe-select",
                        options: function() {
                                return _.map(Places.find().fetch(),
function(object){
                                        return {label: object.country,
value: object.country};
                                });
                        }
                },
        },
        city: {
                type: String,
                label: "Ciudad",
                autoform: {
                        type: "universe-select",
                        options: function() {
                                country = AutoForm.getFieldValue('country');
                                if(!_.isEmpty(country)){
                                        cities = Places.findOne({country:
country}).cities;
                                        var rv = {};

                                        for( var i = 0; i < cities.length;
i++)
                                                rv[i] = cities[i];
                                }

                                return _.map(rv, function(value, idx){
                                        return {label: value, value:
value};
                                });
                        }
                },
        },
        phone: {
                type: String,
                label: "Teléfono",
                optional: true,
                regEx: /^\+?\d{1,3}?[- .]?\(?(?:\d{2,3})\)?[- .]?\d\d\d[-
.]?\d\d\d\d$/,
                autoform: {
                        placeholder: "Ej: +5641521987"
                }
        },
        website: {
                type: String,
                regEx: SimpleSchema.RegEx.Url,
                optional: true,
                autoform: {
                        placeholder: "Direccion virtual. Ej:
http://www.tiendanaturista.cl"
                }
        },
        atention_start: {
                type: String,
                label: "Horario de Apertura",
```

```
                optional: true,
                autoform:{
                        afFieldInput: {
                                type: "time"
                        }
                }
        },
        atention_end: {
                type: String,
                label: "Horario de Cierre",
                optional: true,
                autoform:{
                        afFieldInput: {
                                type: "time"
                        }
                }
        },
        category: {
                type: [String],
                label: "Categoria/s",
                autoform: {
                        type: "universe-select",
                        multiple: true,
                        options: function() {
                                return _.map(Category.find().fetch(),
function(object){
                                        return {label: object.name, value:
object.name};
                                });
                        },
                        placeholder: "Elija una categoria"
                }
        },
        products: {
                type: [String],
                max: 30,
                optional: true,
                label: "Productos: lista de productos separados por una coma.
Ej: pan, ropa, soja, etc",
                autoform: {
                        type: "tags",
                        placeholder: "Una lista de productos que mejorará la
busqueda. Ej: pan, ropa, soja, etc"
                }
        },
        branches: {
                type: [String],
                max: 50,
                optional: true,
                label: "Sucursales"
        },
        author: {
                type: String,
                label: "Autor",
                autoValue: function() {
                        if(this.isInsert)
                                return Meteor.userId();
                        else if(this.isUpsert)
                                        return {$$setOnInsert:
Meteor.userId()};
                                else if(this.isUpdate)
```

```
                                        this.unset();
                },
                autoform: {
                        type: "hidden",
                        label: false
                }
        },
        createdAt: {
                type: Date,
                label: "Fecha de Creación",
                autoValue: function() {
                        if(this.isInsert)
                                return new Date();
                        else if(this.isUpsert)
                                        return {$$setOnInsert: new Date()};
                                else if(this.isUpdate)
                                        this.unset();
                },
                autoform: {
                        type: "hidden",
                        label: false
                }
        },
        commentaries: {
                type: [Commentaries],
                denyUpdate: true,
                optional: true,
                autoform: {
                        type: "hidden"
                }
        },
        ratingSum: {
                type: Number,
                optional: true,
                autoform: {
                        type: "hidden"
                }
        },
        rating: {
                type: Number,
                decimal: true,
                optional: true,
                autoform: {
                        type: "hidden"
                }
        },
        denomination: {
                type: String,
                optional: true,
                autoform: {
                        type: "hidden"
                }
        },
        campaign_rank: {
                type: Number,
                optional: true,
                autoform: {
                        type: "hidden"
                }
        },
        stats_favs: {
```

```
                type: Number,
                optional: true,
                autoform: {
                        type: "hidden"
                }
        },
        stats_visits: {
                type: [Visits],
                optional: true,
                denyUpdate: true,
                autoform: {
                        type: 'hidden'
                }
        }
});

Business.attachSchema(Schemas.Business);

if(Meteor.isClient){
    var hooksObject = {
                onSuccess: function(){
                        Meteor.users.update(
                                Meteor.userId(),
                                {$set: {'profile.full_register': true}}
                        );
                        Bert.alert('Negocio agregado con exito', 'success',
'growl-top-right');

                },
                onError: function(error){
                        Bert.alert('Ha ocurrido un error: '+error.reason,
'danger', 'growl-top-right');
                }
    };

    AutoForm.addHooks('addBusiness', hooksObject);
}
```

- **Promotions.js**

```
Promotions = new Mongo.Collection("promotions");

ImagesPromotions = new FS.Collection("images_promotions", {
                stores: [new FS.Store.FileSystem("images_promotions", {path:
"~/golden_pages_uploads/images/promotions"})],
                filter: {
                        maxSize: 5242880,
                        allow: {
                                contentTypes: ['image/*'],
                        },
                        onInvalid: function (message) {
                                if (Meteor.isClient) {
                                        Bert.alert("El archivo elegido debe
ser una imagen", 'danger', 'growl-top-right');
                                } else {
                                        console.log(message);
                                }
                                }
                }
});
```

```
ImagesPromotions.allow({
    download: function(userId, fileObj) {
        return true;
    }
});

Promotions.allow({
    insert: function(userId, doc){
             return !!userId;
    },
    update: function(userId, doc){
             return !!userId;
    }
});

var Schemas = {};

Schemas.Promotions = new SimpleSchema({
    images: {
             type: String,
             label: "Imagen",
             autoform: {
                      afFieldInput: {
                               type: "fileUpload",
                               collection: "images_promotions",
                               label: "Elija una imagen que represente su
promocion" ,
                               accept: 'image/*'
                      }
             }
    },
    name: {
             type: String,
             label: "Nombre*",
             max: 30,
             autoform: {
                      placeholder: "Ej: Descuento en Glutten"
             }
    },
    desc: {
             type: String,
             label: "Descripcion*",
             max: 140,
             autoform: {
                      afFieldInput: {
                               type: "textarea"
                               },
                      placeholder: "Esta descripcion tiene un maximo de 140
caracteres"
             }
    },
    discount: {
             type: Number,
             label: "Descuento",
             min: 0,
             max: 100,
             optional: true,
             autoform: {
                      placeholder: "Descuento ofrecido a nuestros usuarios.
Entre 0% a 100%"
             }
```

```
        },
        end_date: {
                type: String,
                label: "Fecha de Termino*",
                autoform: {
                        afFieldInput: {
                                type: "bootstrap-datepicker",
                                datePickerOptions: {
                                        language: 'es',
                                        format: "dd/mm/yyyy"
                                }
                        }
                },
                custom: function() {
                        var today = new Date();
                        if(moment(this.value).isBefore(today.value))
                                return "badDate"
                }
        },
        createdAt: {
                type: Date,
                label: "Fecha de Creación",
                autoValue: function() {
                        if(this.isInsert)
                                return new Date();
                        else if(this.isUpsert)
                                        return {$$setOnInsert: new Date()};
                                else if(this.isUpdate)
                                        this.unset();
                },
                autoform: {
                        type: "hidden",
                        label: false
                }
        },
        author: {
                type: String,
                label: "Autor",
                autoValue: function() {
                        return Meteor.userId();
                },
                autoform: {
                        type: "hidden",
                        label: false
                }
        },
        active: {
                type: Boolean,
                optional: true,
                defaultValue: true,
                autoform:{
                        type: "hidden"
                }
        }

});

SimpleSchema.messages({
    badDate: 'La fecha de termino debe ser posterior a hoy.',
});
```

```
Promotions.attachSchema(Schemas.Promotions);

if(Meteor.isClient){
    var hooksObject = {
            onSuccess: function(){
                    Bert.alert('Promocion agregada con exito', 'success',
'growl-top-right');
                    FlowRouter.go('/admin/profile');

            },
            onError: function(error){
                    Bert.alert('Ha ocurrido un error: '+error.reason,
'danger', 'growl-top-right');
            }
    };

    AutoForm.addHooks('addPromotion', hooksObject);
}

Meteor.methods({
    disableToggle: function(id, currentState){
            Promotions.update(id, {
                    $set: {
                            active: !currentState
                    }
            });
    },
    changeEndDate: function (id, date) {
            Promotions.update(id, {
                    $set: {
                            end_date: date
                    }
            });
    }
});
```

- **Tickets.js**

```
Tickets = new Mongo.Collection('tickets');

var Schemas = {};

Messages = new SimpleSchema({
    authorId: {
            type: String,
            autoValue: function() {
                    return Meteor.userId();
            },
            autoform: {
                    type: "hidden",
            }
    },
    author: {
            type: String,
            autoValue: function() {
                    if(Meteor.user()){
                            if(Meteor.user().profile){
                                    if(Meteor.user().profile.name){
                                            return
Meteor.user().profile.name;
                                    }else
```

```
                                                    return
Meteor.user().emails[0].address;
                                }
                                else{
                                                    return
Meteor.user().emails[0].address;
                                }
                        }
                },
                autoform: {
                        type: "hidden",
                }
        },
        createdAt: {
                type: Date,
                autoValue: function() {
                        return new Date()
                },
                autoform: {
                        type: "hidden",
                }
        },
        text: {
                type: String,
                autoform: {
                        type: "hidden"
                }
        }
});

Schemas.Tickets = new SimpleSchema({
        authorId: {
                type: String,
                autoValue: function() {
                        if(this.isInsert)
                                return Meteor.userId();
                        else if(this.isUpsert)
                                                    return {$$setOnInsert:
Meteor.userId()};
                                else if(this.isUpdate)
                                                    this.unset();
                },
                autoform: {
                        type: "hidden",
                }
        },
        createdAt: {
                type: Date,
                label: "Fecha de Creación",
                autoValue: function() {
                        if(this.isInsert)
                                return new Date();
                        else if(this.isUpsert)
                                                    return {$$setOnInsert: new Date()};
                                else if(this.isUpdate)
                                                    this.unset();
                },
                autoform: {
                        type: "hidden",
                        label: false
                }
```

```
        },
        type: {
                type: String,
                allowedValues: ['bugs', 'business', 'page'],
                autoform: {
                        type: "hidden"
                }
        },
        support_read: {
                type: Boolean,
                defaultValue: false,
                autoform: {
                        type: "hidden"
                }
        },
        author_read: {
                type: Boolean,
                defaultValue: true,
                autoform: {
                        type: "hidden"
                }
        },
        subject: {
                type: String,
                max: 30,
                autoform: {
                        type: "hidden"
                }
        },
        messages: {
                type: [Messages],
                denyUpdate: true,
                autoform: {
                        type: "hidden"
                }
        }
    });

    Tickets.attachSchema(Schemas.Tickets);
```

- **Visits.js**
  La razón de ser de este pequeño código es que la aplicación tenga una referencia a la colección en MongoDB y pueda ser capaz de usar la API necesaria.

```
    Visits = new Mongo.Collection("visits");
```

#### 6.4.1.4 FlowRouter

Esta es la forma que tiene Meteor para redirigir al usuario a diferentes pantallas sin recargar la aplicación y permite la carga dinámica de elementos. Para esto se utilizan rutas en vez de links comunes del html.

- **Rutes.js**

```
    if(Meteor.isClient){
        Accounts.onLogin(function(){
                if(Meteor.user().roles == null){
                        Meteor.call("defaultRoleSet");
                }
```

```
                FlowRouter.go('home');
        });

        Accounts.onLogout(function(){
                FlowRouter.go('home');
        });
}

FlowRouter.triggers.enter(
        [function(context, redirect){
                if(!Meteor.userId())
                        FlowRouter.go('home');
        }],
        {except: ["login", "business"]}
);

FlowRouter.route('/', {
        name: 'home',
        action() {
                BlazeLayout.render('MainLayout', {main: 'HomeLayout'});
        }
});

FlowRouter.route('/login', {
        name: 'login',
        action() {
                BlazeLayout.render('MainLayout', {main: 'LogInLayout'});
        }
});

FlowRouter.route('/addbusiness', {
        name: 'addbusiness',
        action() {
                BlazeLayout.render('MainLayout', {main: 'addBusinessLayout'});
        }
});

FlowRouter.route('/business/:id', {
        name: 'business',
        action() {
                BlazeLayout.render('MainLayout', {main: 'BusinessSingle'});
        }
});

FlowRouter.route('/createaccount', {
        name: 'createaccount',
        action() {
                BlazeLayout.render('MainLayout', {main: 'createAccount'});
        }
});

FlowRouter.route('/support', {
        name: 'support',
        action() {
                BlazeLayout.render('MainLayout', {main: 'Support'});
        }
});

//Dashboard
FlowRouter.route('/admin', {
        name: 'admin',
```

```
        action() {
                BlazeLayout.render('DashboardLayout', {main: 'Dashboard'});
        }
});

FlowRouter.route('/ticket/:id', {
        name: 'ticket',
        action() {
                BlazeLayout.render('DashboardLayout', {main: 'TicketSingle'});
        }
});

var adminRoutes = FlowRouter.group({
        prefix: '/admin',
        name: 'admin'
});

adminRoutes.route('/users', {
        name: 'users',
        action() {
                BlazeLayout.render('DashboardLayout', {main: 'Users'});
        }
});

adminRoutes.route('/profile', {
        name: 'profile',
        action() {
                BlazeLayout.render('DashboardLayout', {main: 'Profile'});
        }
});

adminRoutes.route('/changepassword', {
        name: 'changepassword',
        action() {
                BlazeLayout.render('DashboardLayout', {main:
'changePassword'});
        }
});

adminRoutes.route('/changeemail', {
        name: 'changeemail',
        action() {
                BlazeLayout.render('DashboardLayout', {main: 'changeEmail'});
        }
});

adminRoutes.route('/businessupdate', {
        name: 'businessupdate',
        action() {
                BlazeLayout.render('DashboardLayout', {main:
'businessUpdate'});
        }
});

adminRoutes.route('/promotionupdate', {
        name: 'promotionupdate',
        action() {
                BlazeLayout.render('DashboardLayout', {main:
'promotionUpdate'});
        }
});
```

```
adminRoutes.route('/addpromotion', {
    name: 'addpromotion',
    action() {
            BlazeLayout.render('DashboardLayout', {main: 'addPromotion'});
    }
});

adminRoutes.route('/businesses', {
    name: 'businesses',
    action() {
            BlazeLayout.render('DashboardLayout', {main:
'adminBusinesses'});
    }
});

adminRoutes.route('/tickets', {
    name: 'tickets',
    action() {
            BlazeLayout.render('DashboardLayout', {main: 'adminTickets'});
    }
});

FlowRouter.route('/favorites', {
    name: 'favorites',
    action() {
            BlazeLayout.render('DashboardLayout', {main: 'Favorites'});
    }
});
```

#### 6.4.1.5 Layouts

- **DashboardLayout.html**

```
<template name="DashboardLayout">
    {{> menuBar}}
    {{> appNav}}
            <main class="dashboard-layout">
                    {{> Template.dynamic template=main}}
            </main>
</template>
```

- **HomeLayout.html**

```
<template name="HomeLayout">
  {{> mainSlider}}
  {{> searchBar}}
  <div class="container">
    <div class="row">
      <div class="result col-sm-10 col-sm-offset-1" id="result">
        {{#each Businesses}}
          {{> Business}}
        {{/each}}
        <div class="col-sm-12">
          <ul class="pager">
              <li class="previous"><a href="#result"><span aria-
hidden="true">&larr;</span> Anterior</a></li>
              <li class="next"><a href="#result">Siguiente <span aria-
hidden="true">&rarr;</span></a></li>
          </ul>
        </div>
      </div>
    </div>
  </div>
</template>
```

- **HomeLayout.js**

```
Template.HomeLayout.helpers({
   Businesses: function(){
            keyword = Session.get('search/keyword');
            skipCount = Session.get('skipCount');

            Deps.autorun(function(){
                     Meteor.subscribe("search", keyword, skipCount);
            });

            var and = [];

            if(Session.get('country')){
                     and.push( { country: Session.get('country')} );
            }

            if(Session.get('city')){
                     and.push( { city: Session.get('city')} );
            }

            if(Session.get('category')){
                     and.push( { category: Session.get('category')} );
            }

            if(!keyword){
                     if(and && and.length != 0){
                              return Business.find({$and: and}, {limit: 6,
skip: skipCount});
                     }
                     else{
                              return Business.find({}, {limit: 6, skip:
skipCount});
                     }
            }else{
                     if(and && and.length != 0){
                              return Business.find({$and: and}, {sort:
[["score", "desc"]], limit: 6, skip: skipCount});
```

```
                        }
                        else{
                                return Business.find({}, {sort: [["score",
"desc"]], limit: 6, skip: skipCount});
                        }
                }
        }
});

Template.HomeLayout.events({
        'click .next': function(){

        if(Template.HomeLayout.__helpers.get('Businesses').call().count() != 0){
                        Session.set('skipCount', Session.get('skipCount') +
6);
                }
        },
        'click .previous': function(){
                if(Session.get('skipCount') > 0){
                        Session.set('skipCount', Session.get('skipCount') -
6);
                }
        }
});

Template.HomeLayout.onRendered(function(){
        Session.setDefault("skipCount", 0);
        Session.setDefault('search/keyword', "");

        if(!Session.get('visitTracked')){
                Meteor.call('trackSiteVisits');
                Session.set('visitTracked', true);
        }
});
```

- **LogInLayout.js**

```
T9n.setLanguage('es');//en,es,pt
AccountsTemplates.configure({
        negativeValidation: true,
        showValidating: true,
        forbidClientAccountCreation: false
});
```

- **MainLayout.html**

  En la línea 4 podemos ver una sección donde Blaze carga dinámicamente el contenido de este Layout.

```
<template name="MainLayout">
        <body>
                        {{> menuBar}}
                        {{> Template.dynamic template=main}}
        </body>
</template>
```

- **addBusinessLayout.html**

```
<template name="addBusinessLayout">
    {{#if isInRole 'business'}}
            {{#if full_register}}
                    <div class="container">
                            <h1>Registro Completado</h1>
                    </div>
            {{else}}
                    <div class="container" style="margin-top: 20px">
                            <h2>Agregar Negocio</h2>
                            {{> quickForm
                                    collection="Business"
                                    id="addBusiness"
                                    type="insert"
                            }}
                    </div>
            {{/if}}
    {{else}}
            {{> unauthorized}}
    {{/if}}
</template>
```

- **addBusinessLayout.js**

```
Template.addBusinessLayout.helpers({
    full_register: function(){
            if(Meteor.user()){
                    if(Meteor.user().profile){
                            if(Meteor.user().profile.full_register){
                                    return
Meteor.user().profile.full_register;
                            }
                    }else{
                            return false;
                    }
            }
    }
});
```

#### 6.4.1.6 Componentes

Estos componentes son trozos de código reutilizables y mantenibles al ser almacenados separadamente del código principal. Un componente viene a ser una parte pequeña de la aplicación como barras de menú, secciones, etc.

- **Busines.html**

```
<template name="Business">
  <div class="container col-md-4 col-md-offset-0 col-sm-6 col-sm-
offset-0 col-xs-10 col-xs-offset-1">
    <div class="storeItem">
      <div class="storeImage">
        <a href="{{Images.url}}" target="_blank"><img
src="{{Images.url}}" alt=""></a>
      </div>
      <div class="container-fluid storeInfo">
            <h4>{{name}}</h4>
            <h6>{{subtitle}}</h6>
            <p>{{desc}}</p>
        <h6>{{address}}</h6>
        <h6>Horario de atenciòn:<br>{{atention_start}} a
{{atention_end}}</h6>
      </div>
      <div class="container-fluid storeLink">
      {{#if currentUser}}
      <button class="btn col-xs-offset-5 col-sm-offset-3 col-md-offset-
1 col-lg-offset-4 {{#if inFav}}removefav btn-danger{{else}}addfav btn-
primary{{/if}}"><i class="{{#if inFav}}fa fa-times{{else}}fa fa-
star{{/if}}"></i></button>
      {{/if}}
            <a href='/business/{{_id}}' class="btn btn-primary">Mas
Informacion</a>
      </div>
    </div>
  </div>
</template>
```

- **Business.js**

```
Template.Business.helpers({
    Images: function(){
            return Images.findOne(this.images[0]);
    },
    inFav: function(){
            var favIds = _.pluck(Meteor.user().favorites, "id");
            return favIds.includes(this._id);
    }
});

Template.Business.events({
    'click .addfav':function(event){
            Meteor.call('addFav', Meteor.userId(), this._id,
this.name);
    },
    'click .removefav':function(event){
            Meteor.call('removeFav', Meteor.userId(), this._id);
    }
});
```

- **Promotions.html**

```
<template name="Promotion">
    <div class="container promotionItem">
            <div class="container-fluid col-md-8">
                    <h3>{{this.name}}</h3>
                    <p>{{this.desc}}</p>
                    <h4>Descuento</h4>
                    <strong><h5>{{this.discount}}%</h5></strong>
                    <p>Promocion disponible hasta el:
{{p_endDate}}</p>
                    <a href="/admin/promotionupdate" class="edit btn
btn-primary col-md-4"><i class="fa fa-pencil"></i> Modificar datos de
Promocion</a>
                    <button class="remove btn btn-danger col-md-4"><i
class="fa fa-times"></i> Eliminar Promocion</button>
                    <button class="{{#if
active}}active_promotion{{else}}disabled_promotion{{/if}} disable-
toggle btn btn-success col-md-4"><i class="fa fa-toggle-on"></i>
Activa</button>
                    <button class="{{#if
active}}disabled_promotion{{else}}active_promotion{{/if}} disable-
toggle btn btn-default col-md-4"><i class="fa fa-toggle-off"></i>
Desabilitada</button>
            </div>
            <div class="container-fluid promotionImage col-md-4">
                    <a href="{{p_images.url}}" target="_blank"><img
src="{{p_images.url}}" alt=""></a>
            </div>
    </div>
</template>
```

- **Promotions.js**

```
Template.Promotion.helpers({
    p_endDate: function() {
            return moment(this.end_date).format('DD/MM/YYYY');
    },
    p_images: function(){
                    return ImagesPromotions.findOne(this.images);
    }
});

Template.Promotion.events({
    'click .edit': function(){
            Session.set('promotionId', this._id);
    },
    'click .remove': function(){
            removed = ImagesPromotions.remove(this.images);
            if(removed == 1)
                    Promotions.remove(this._id);
    },
    'click .disable-toggle': function(){
            if(this.active == false){
                    var id = this._id;
                    var active = this.active;

                    bootbox.prompt({
                            size: "small",
                            title: "Nueva fecha de termino para la
promoción",
                            inputType: "date",
```

```
                                    callback: function(result){
                                              var today = new Date();
                                              if(result){

        if(moment(result).isBefore(today.value)){

        Bert.alert('La nueva fecha debe ser posterior al dia de hoy',
    'danger', 'growl-top-right');
                                                        }else{

        Meteor.call('changeEndDate', id, result);

        Meteor.call('disableToggle', id, active);
                                                        }
                                              }
                                    },
                                    closeButton: false
                          });
                }else{
                          Meteor.call('disableToggle', this._id,
    this.active);
                }
        }
    });
```

- **PromotionsForUsers.html**

```
    <template name="PromotionsForUsers">
    <div class="container promotionItem">
    <div class="container-fluid col-md-8">
    <h3>{{this.name}}</h3>
    <p>{{this.desc}}</p>
    <h4>Descuento</h4>
    <strong><h5>{{this.discount}}%</h5></strong>
    <p>Promocion disponible hasta el: {{p_endDate}}</p>
    </div>
    <div class="container-fluid promotionImage col-md-4">
    <a href="{{p_images.url}}" target="_blank"><img src="{{p_images.url}}"
    alt=""></a>
    </div>
    </div>
    </template>
```

- **PromotionsForUsers.js**

```
    Template.PromotionsForUsers.helpers({
        p_endDate: function() {
                  return moment(this.end_date).format('DD/MM/YYYY');
        },
        p_images: function(){
                          return ImagesPromotions.findOne(this.images);
        }
    });
```

- **appNav.html**

```
<template name="appNav">
    <div class="app-nav">
        <ul>
            <li><a href="/admin"><i class="fa fa-dashboard"></i>
<span>Tablero</span></a></li>
            <li><a href="/admin/profile"><i class="fa fa-user"></i>
<span>Perfil</span></a></li>
            <li><a href="/favorites"><i class="fa fa-star"></i>
<span>Favoritos</span></a></li>
            <li><a href="/admin/tickets"><i class="fa fa-
envelope"></i><span>Solicitudes de Soporte</span></a></li>
            {{#if isInRole 'super-admin, minister'}}
            <li><a href="/admin/users"><i class="fa fa-users"></i>
<span>Manejo de Usuarios</span></a></li>
            <li><a href="/admin/businesses"><i class="fa fa-
building"></i><span>Manejo de Negocios</span></a></li>
            {{/if}}
        </ul>
    </div>
</template>
```

- **businessComments.html**

```
<template name="businessComments">
   <div class="container">
            <div class="insertComment">
                    <h3>Comentarios</h3>
                    <form>
                            <div class="form-group">
                                    <label>Calificación</label>
                                    {{> starsRating
id='commentRating' mutable=true size=30}}
                            </div>
                            <div class="form-group">
                                    <label>Agregue un
comentario</label>
                                    <textarea  class="form-control"
id="comment" rows="4" placeholder="Aqui puede incluir un comentario de
forma opcional"></textarea>
                            </div>
                            <button class="btn btn-default">Enviar
Comentario</button>
                    </form>
                    <br>
            </div>
   </div>
</template>
```

- **businessComments.js**

```
Template.businessComments.events({
    'submit form': function(event){
                event.preventDefault();
                rating = $('#commentRating').data('userrating');
                if(!rating){
                        Bert.alert("Debe agregar una calificación de 1 a
5", 'danger', 'growl-top-right');
                }else{
                        businessId = FlowRouter.getParam("id");
                        Business.update({_id: businessId},
                                {
                                        $inc: {
                                                ratingSum : rating
                                        },
                                        $push: {
                                                commentaries: {
                                                        rating:
rating,
                                                        comment:
event.target.comment.value
                                                }
                                        }
                                }
                        );

                        updatedBusiness = Business.findOne(businessId);
                        newRating = updatedBusiness.ratingSum /
updatedBusiness.commentaries.length;
                        Business.update({_id: businessId},
                                {
                                        $set: {
                                                rating: newRating
                                        }
                                }
                        );
                        event.target.comment.value = "";
                        $('#commentRating').trigger('reset');
                }
    }
});
```

- **createAccount.html**

```
<template name="createAccount">
    <div class="container">
            {{#if isInRole 'super-admin, minister'}}
                    <h2 class="col-sm-offset-4">Crear nueva
cuenta</h2>
                    <form class="registerAccount form-horizontal
form-signin">
                            <div class="form-group">
                                    <label for="inputEmail"
class="col-sm-4 control-label" >Email:</label>
                                    <div class="col-sm-8">
                                            <input type="email"
class="form-control" id="inputEmail" name="inputEmail"
placeholder="Email">
                                    </div>
                            </div>
                            <div class="form-group">
                                    <label for="inputPassword"
class="col-sm-4 control-label" >Contraseña:</label>
                                    <div class="col-sm-8">
                                            <input type="text"
class="form-control" id="inputPassword" name="inputPassword"
placeholder="Contraseña" value={{randomPassword}}>
                                    </div>
                            </div>
                            <div class="form-group">
                                    <label for="selectRole"
class="col-sm-4 control-label" >Tipo de Cuenta:</label>
                                    <div class="col-sm-8">
                                            <select class="form-
control" id="selectRole" name="selectRole">
                                                    <option
value="" disabled selected>Elija un tipo de Cuenta</option>
                                                    <option
value="minister">Ministro</option>
                                                    <option
value="business">Negocio</option>
                                            </select>
                                    </div>
                            </div>
                            <div class="form-group">
                                    <div class="col-sm-offset-4
col-sm-8">
                                            <button type="submit"
class="btn btn-primary">Crear</button>
                                    </div>
                            </div>
                    </form>
            {{else}}
                    {{> unauthorized}}
            {{/if}}
    </div>
</template>
```

- **createAccount.js**

```
Template.createAccount.helpers({
    randomPassword: function(){
            return Random.secret(6);
    }
});
Template.createAccount.events({
    'submit form': function(event) {
                    event.preventDefault();
                    var emailVar = event.target.inputEmail.value;
                    var passwordVar =
event.target.inputPassword.value;
                    var userRole = event.target.selectRole.value;
                    error = Meteor.call("createUsers", emailVar,
passwordVar, userRole, function(error){
                                if(error){
                                        Bert.alert(error.reason,
'danger', 'growl-top-right');
                                }else{
                                Bert.alert('Cuenta creada con exito',
'success', 'growl-top-right');
                                }
                    });
            }
});

Template.createAccount.onRendered(function(){
    $('.registerAccount').validate({
            rules: {
                    inputEmail: {
                            required: true,
                            email: true
                    },
                    inputPassword: {
                            required: true,
                            minlength: 6
                    },
                    selectRole: {
                            required: true
                    }
            },
            messages: {
                    inputEmail: {
                            required: "Una direccion de correo es
obligatoria.",
                            email: "Ha ingresado una direccion de
correo invalida"
                    },
                    inputPassword: {
                            required: "Una contraseña es
obligatoria",
                            minlength: "Su contraseña debe tener al
menos {0} caracteres"
                    },
                    selectRole: {
                            required: "Debe seleccionar un tipo de
cuenta"
                    }
            }
    });
});
```

- **mainSlider.html**

```
<template name="mainSlider">
  <div id="carousel-example-generic" class="carousel slide" data-
ride="carousel">
    <!-- Indicators -->
    <ol class="carousel-indicators">
      <li data-target="#carousel-example-generic" data-slide-to="0"
class="active"></li>
      <li data-target="#carousel-example-generic" data-slide-
to="1"></li>
    </ol>

    <!-- Wrapper for slides -->
    <div class="carousel-inner" role="listbox">
      <div class="item active">
        <img src="/images/slider/food-q-c-1600-500-10.jpg" alt="...">
        <div class="carousel-caption">
          <h3>Pan</h3>
          <p>El pan es muy rico</p>
        </div>
      </div>
      <div class="item">
        <img src="/images/slider/food-q-c-1600-500-6.jpg" alt="...">
        <div class="carousel-caption">
        </div>
      </div>
    </div>

    <!-- Controls -->
    <a class="left carousel-control" href="#carousel-example-generic"
role="button" data-slide="prev">
      <span class="glyphicon glyphicon-chevron-left" aria-
hidden="true"></span>
      <span class="sr-only">Anterior</span>
    </a>
    <a class="right carousel-control" href="#carousel-example-generic"
role="button" data-slide="next">
      <span class="glyphicon glyphicon-chevron-right" aria-
hidden="true"></span>
      <span class="sr-only">Siguiente</span>
    </a>
  </div>
</template>
```

- **menuBar.html**

```
<template name="menuBar">
  <nav class="navbar navbar-default">
  <div class="container-fluid">
    <!-- Brand and toggle get grouped for better mobile display -->
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-
toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-
expanded="false">
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="/"><img src="/images/revologo.png"
alt=""></a>
    </div>

    <!-- Collect the nav links, forms, and other content for toggling -
->
    <div class="collapse navbar-collapse" id="bs-example-navbar-
collapse-1">
      <ul class="nav navbar-nav">
        <li class="active"><a href="/">Inicio<span class="sr-
only">(current)</span></a></li>
        <li><a href="#">Obtenga su pasaporte</a></li>
        <li><a href="#">Sobre nosotros</a></li>
      </ul>
      <ul class="nav navbar-nav navbar-right">
        {{#if currentUser}}
        <li><a href="/support">Soporte</a></li>
          <li class="dropdown">
          <a href="#" class="dropdown-toggle" data-toggle="dropdown"
role="button" aria-haspopup="true" aria-expanded="false">{{user}}<span
class="caret"></span></a>
          <ul class="dropdown-menu">
            <li><a href="/admin" id="dash">Tablero</a></li>
            <li><a href="/admin/profile" id="dash">Perfil</a></li>
            <li><a href="/favorites" id="favs">Favoritos</a></li>
            <li><a href="#" id="logout">Salir</a></li>
          </ul>
        </li>
        {{else}}
          <li><a href="/login">Ingresar</a></li>
        {{/if}}
        <li class="dropdown">
          <a href="#" class="dropdown-toggle" data-toggle="dropdown"
role="button" aria-haspopup="true" aria-expanded="false">Idioma<span
class="caret"></span></a>
          <ul class="dropdown-menu">
            <li><a href="#">Español</a></li>
            <li><a href="#">Ingles</a></li>
            <li><a href="#">Portugués</a></li>
          </ul>
        </li>
      </ul>
    </div><!-- /.navbar-collapse -->
  </div><!-- /.container-fluid -->
</nav>
</template>
```

- **mainBar.js**

```
Template.menuBar.helpers({
    user: function() {
              if(Meteor.user()){
                       if(Meteor.user().profile){
                                 if(Meteor.user().profile.name){
                                          return
Meteor.user().profile.name;
                                 }else
                                          return
Meteor.user().emails[0].address;
                       }
                       else{
                                 return Meteor.user().emails[0].address;

                       }
              }
    }
});

Template.menuBar.events({
    'click #logout': function(){
              Meteor.logout();
    }
});
```

- **searchBar.html**

```
<template name="searchBar">
  <div class="container searchBar">
    <div class="row">
      <div class="col-sm-10 col-sm-offset-1">
        <div class="well">
          <div style="overflow: hidden;">
            <form class="searchBar">
              <div class="col-sm-12">
                <div class="form-group">
                  <div class="input-group">
                    <input type="text" class="form-control searchInput"
placeholder="Busca Tienda, Producto, etc">
                    <span class="input-group-btn">
                      <button class="btn btn-default"
type="submit"><span class="fa fa-search"></span></button>
                    </span>
                  </div>
                </div>
              </div>
              <div class="form-group">
                <div class="col-sm-4 col-md-4">
                  <select id="countrySelect" class="form-control">
                    <option value="" selected disabled>Pais</option>
                    <option value="">Todos</option>
                    {{#each country}}
                                 <option
value="{{this.country}}">{{this.country}}</option>
                    {{/each}}
                  </select>
                </div>
              </div>
              <div class="form-group">
                <div class="col-sm-4 col-md-4">
```

```
                    <select id="citySelect" class="form-control">
                      <option value="" selected disabled>Ciudad</option>
                      <option value="">Cualquiera</option>
                      {{#each city}}
                                        <option
value="{{this}}">{{this}}</option>
                      {{/each}}
                    </select>
                  </div>
                </div>
                <div class="form-group">
                  <div class="col-sm-4 col-md-4">
                    <select id="categorySelect" class="form-control">
                      <option value="" selected
disabled>Categoria</option>
                      <option value="">Todas</option>
                      {{#each category}}
                                         <option
value="{{this.name}}">{{this.name}}</option>
                      {{/each}}
                    </select>
                  </div>
                </div>
              </form>
            </div>
          </div>
        </div>
      </div>
    </template>
```

- **searchBar.js**

```
Template.searchBar.onRendered(function(){
    $('select').select2();
});

Template.searchBar.events({
    'change #countrySelect'(event){
              Session.set('country', event.target.value);
              Session.set('city', "");
    },
    'change #citySelect'(event){
              Session.set('city', event.target.value);
    },
    'change #categorySelect'(event){
              Session.set('category', event.target.value);
    },
    'keyup .searchInput'(event){
              Session.set('search/keyword', event.target.value);
    },
    'submit .searchBar'(event){
              event.preventDefault();
    }
});
```

```
Template.searchBar.helpers({
    country: function(){
            return Places.find().fetch();
    },
    city: function(){
            var cities = Places.findOne({country:
Session.get('country')});
            return cities && cities.cities.sort();
    },
    category: function(){
            return Category.find().fetch().sort();
    }
});
```

#### 6.4.1.7 Paginas

Estas paginas son pantallas completas, mostrables para el usuario y que hacen uso de los componentes y se renderizan dentro de las plantillas o layouts enseñados anteriormente.

- **BusinessSingle.html**

```
<template name="BusinessSingle">
    <div class="container businessDetails">


            <div class="businessInfo col-sm-6">

                    <h1>{{businessSingle.name}}</h1>
                    <h4>{{businessSingle.subtitle}}</h4>
                    <br>
                    <h4>Dirección</h4>

    <p><strong>{{businessSingle.address}}</strong></p>
                    <p><strong>{{businessSingle.city}} -
{{businessSingle.country}}</strong></p>
                    {{#if businessSingle.phone}}
                            <h4>Teléfono</h4>
                            <p>{{businessSingle.phone}}</p>
                    {{/if}}
                    {{#if businessSingle.website}}
                            <h4>Pagina Web</h4>
                            <a
href="{{businessSingle.website}}">{{businessSingle.website}}</a>

                    {{/if}}
                    <h4>Horario de Atención</h4>
                    <p><strong>{{businessSingle.atention_start}} -
{{businessSingle.atention_end}}</strong></p>
                    <h4>Descuento</h4>

    <h4><strong>{{businessSingle.discount}}%</strong></h4>

                    {{#if currentUser}}
                            {{#if inFav}}
                                    <button class="btn removefav
btn-danger"><i class="fa fa-times"></i> Quitar de Favoritos</button>
                            {{else}}
                                    <button class="btn addfav btn-
primary"><i class="fa fa-star"></i> Agregar a Favoritos</button>
                            {{/if}}
```

```
                        {{/if}}
                        <h4>Descripción</h4>
                        <p>{{businessSingle.desc}}</p>
                        <h4>Categoria/s</h4>
                        <ul>
                                {{#each businessSingle.category}}
                                        <li>{{this}}</li>
                                {{/each}}
                        </ul>
                        <h4>Productos</h4>
                        <ul>
                                {{#each businessSingle.products}}
                                        <li>{{this}}</li>
                                {{/each}}
                        </ul>
                        {{#if businessSingle.branches}}
                                <h4>Sucursales</h4>
                                        <ul>
                                                {{#each
businessSingle.branches}}

    <li><p><strong>{{this}}</strong></p></li>
                                                {{/each}}
                                        </ul>
                         {{/if}}
                </div>
                <div class="images col-md-6">
                <br><br><br><br><br><br>
                        <div id="carousel-example-generic"
class="carousel slide" data-ride="carousel">
                                <!-- Indicators -->
                                <ol class="carousel-indicators">
                                        <li data-target="#carousel-
example-generic" data-slide-to="0" class="active"></li>
                                        <li data-target="#carousel-
example-generic" data-slide-to="1"></li>
                                </ol>

                                <!-- Wrapper for slides -->
                                <div class="carousel-inner"
role="listbox">
                                        {{#each businessImages}}
                                                {{> carouselItem
imageUrl=url index=@index}}
                                        {{/each}}
                                </div>

                                <!-- Controls -->
                                <a class="left carousel-control"
href="#carousel-example-generic" role="button" data-slide="prev">
                                        <span class="glyphicon
glyphicon-chevron-left" aria-hidden="true"></span>
                                        <span class="sr-
only">Anterior</span>
                                </a>
                                <a class="right carousel-control"
href="#carousel-example-generic" role="button" data-slide="next">
                                        <span class="glyphicon
glyphicon-chevron-right" aria-hidden="true"></span>
                                        <span class="sr-
only">Siguiente</span>
```

```
                                    </a>
                            </div>
                            <div class="ratings">

        <strong><h4>{{businessSingle.denomination}} <span>
                                                {{> starsRating
    rating=businessSingle.rating size=30 id='rating'}}
                                                {{> starsRating
    rating=businessSingle.campaign_rank size=30 id='rank' class='awesome'
    star='\\f1b1'}}
                                    </span></h4></strong>
                                    <p>Las cucharas representan el estandar
    de exelencia que tiene el Negocio.</p>
                            </div>
                    </div>
        </div>

        <div class="container promotions">
                    <h3>Promociones</h3>
                    {{#each promotions}}
                            {{> PromotionsForUsers}}
                    {{/each}}
        </div>
        {{#if currentUser}}
                    <div class="container insertComments">
                            {{> businessComments}}
                    </div>
        {{/if}}
        {{#if businessSingle.commentaries}}
                    <div class="container commentaries-section">
                            <h4>Comentarios anteriores</h4>
                            <div class="container commentaries">
                                    {{#each businessSingle.commentaries}}
                                            {{> starsRating
    rating=this.rating}}

        <strong>{{this.author}}</strong>
                                            <p>{{this.comment}}</p>
                                    {{/each}}
                            </div>
                    </div>
        {{/if}}
    </template>

    <template name="carouselItem">
        <div class="item {{isActive}}">
                    <img src="{{imageUrl}}">
        </div>
    </template>
```

- **BusinessSingle.js**

```
Template.BusinessSingle.helpers({
    businessSingle: function(){
            var id = FlowRouter.getParam("id");
            return Business.findOne(id);
    },
    businessImages: function(){
            return
Template.BusinessSingle.__helpers.get('businessSingle').call() &&
Images.find({_id: {$in:
Template.BusinessSingle.__helpers.get('businessSingle').call().images}}
);
    },
    inFav: function(){
            var favIds = [];
            if(Meteor.user()){
                    favIds = _.pluck(Meteor.user().favorites, "id");
            }
            return favIds.includes(FlowRouter.getParam("id"));
    },
    promotions: function(){
            return
Template.BusinessSingle.__helpers.get('businessSingle').call() &&
Promotions.find({author:
Template.BusinessSingle.__helpers.get('businessSingle').call().author})
;
    }
});

Template.BusinessSingle.events({
    'click .addfav':function(event){
            Meteor.call('addFav', Meteor.userId(),
FlowRouter.getParam("id"),
Template.BusinessSingle.__helpers.get('businessSingle').call().name);
    },
    'click .removefav':function(event){
            Meteor.call('removeFav', Meteor.userId(),
FlowRouter.getParam("id"));
    }
});

Template.BusinessSingle.onRendered(function() {
    Meteor.call('trackVisit', FlowRouter.getParam("id"));

    var $stars = $('.awesome');
    switch($stars.data().rating){
            case 1:
                    var text = "<strong>1 Cuchara:</strong><br>Acepta
el pasaporte y pegar okis en su local.";
                    break;
            case 2:
                    var text = "<strong>2
Cucharas:</strogn><br>Distribuye información.<br>- Distribuir okis
pequeños y grandes.<br>- Tener sello oki de su establecimiento.<br>-
Recoger correos.";
                    break;
            case 3:
                    var text = "<strong>3 Cucharas:</strong><br>Tiene
materiales del Arsenal.<br>- Distribuir pasaportes.<br>- Haber
adquirido el material básico para la distribución.";
                    break;
```

```
                case 4:
                        var text = "<strogn>4
Cucharas:</strong><br>Realiza actividades.<br>- Condecora embajadores y
ministros.<br>- Distribuye cierta cantidad de material.<br>- Realiza un
evento dentro del transcurso de 3 meses.<br>- Tiene pendones de la
recolución de la cuchara en sus paredes.<br>- Afiches de oki
enmarcados.<br>- Decora mesas con okis.";
                        break;
                case 5:
                        var text = "<strong>5 Cucharas:</strong>
Ministerio muy activo en todas las áreas:<br>- Abre nuevos
ministerios.<br>- Hace reuniones de activistas en su local.<br>-
Organiza eventos o ferias regularmente.<br>-Es activo en generar nuevas
ideas para expandir la campaña.";
                        break;

    }
    $stars.attr('title', text).data('toggle',
'tooltip').data('placement', 'left').data('html', true).tooltip();
});

Template.carouselItem.helpers({
    isActive: function () {
                return (this.index === 0) ? 'active': '';
    }
});
```

- **BusinessUpdate.html**

```
<template name="businessUpdate">
    {{> quickForm
                collection="Business"
                id="updateBusinessForm"
                type="update"
                doc=business
    }}
</template>
```

- **BusinessUpdate.js**

```
Template.businessUpdate.helpers({
    business: function(){
                return Business.findOne({author: Meteor.userId()});
    }
});
```

- **Dashboard**

```
<template name="Dashboard">
    <h1>Tablero</h1>
    <h2>Estadisticas Mensuales</h2>
    {{#if isInRole 'business'}}
    <div class="container stats stats-favs">
                <h3>Favoritos</h3>
                <p>{{business.stats_favs}}</p>
    </div>
    <div class="container stats stats-visits">
                <h3>Visitas Mensuales</h3>
                <p>{{businessMonthlyVisits}}</p>
    </div>
    {{/if}}
```

```
        {{#if isInRole 'minister'}}
                <div class="container stats stats-businesses">
                        <h3>Actividad de Negocios</h3>
                <table class="table table-hover">
                <thead>
                    <tr>
                        <th>ID</th>
                        <th>Nombre</th>
                        <th>Último inicio de sesión</th>
                        <th>Estado</th>
                        <th>Visitas</th>
                    </tr>
                </thead>
                <tbody>
                    {{#each ministerBusinesses}}
                                {{> ministerBusinessesTr}}
                    {{/each}}
                </tbody>
                </table>
                 </div>
        {{/if}}
        {{#if isInRole 'super-admin'}}
                <div class="container stats stats-site_visits">
                            <h3>Visitas al sitio</h3>
                            <p>{{monthlySiteVisits}}</p>
                </div>
                 <div class="container stats stats-normal-users">
                        <h3>Embajadores Activos</h3>
                        <p>{{activeNormalUsers}} de un total de
    {{totalNormalUsers}}</p>
                 </div>
                 <div class="container stats stats-businesses">
                        <h3>Negocios Activos</h3>
                        <p>{{activeBusinesses}} de un total de
    {{totalBbusinesses}}</p>
                 </div>
        {{/if}}
    </template>

    <template name="ministerBusinessesTr">
                <tr>
                            <td>{{_id}}</td>
                            <td>{{name}}</td>
                            <td>{{lastLogin}}</td>
                            <td>{{status}}</td>
                            <td>{{monthlyVisits}}</td>
                </tr>
    </template>
```

- **Dashboard.js**

```
    Template.Dashboard.helpers({
        business: function(){
                return Business.findOne({author: Meteor.userId()});
        },
        businessMonthlyVisits: function(){
                visitsArray = Business.findOne({author: Meteor.userId()});
                if(!visitsArray)
                        return 0;
                visitsArray = visitsArray.stats_visits;
                date = new Date();
```

```
                for(var i = 0; i < visitsArray.length; i++){
                        if(visitsArray[i].month == date.getMonth() &&
visitsArray[i].year == date.getFullYear()){
                                return visitsArray[i].count;
                        }
                }

        },
        activeNormalUsers: function(){
                // crear indice para usuarios segun roles
                normalUsers = Meteor.users.find({roles: "normal-
user"}).fetch();
                aMonthLater = new Date();
                aMonthLater.setMonth(aMonthLater.getMonth() - 1);

                for(var i = 0, count = 0; i < normalUsers.length; i++){
                        if(normalUsers[i].status.lastLogin.date.getTime()
> aMonthLater.getTime()){
                                count++;
                        }
                }

                return count;
        },
        totalNormalUsers: function(){
                return Meteor.users.find({roles: "normal-user"}).count();
        },
        activeBusinesses: function(){
                businesses = Meteor.users.find({roles:
"business"}).fetch();
                aMonthLater = new Date();
                aMonthLater.setMonth(aMonthLater.getMonth() - 1);

                for(var i = 0, count = 0; i < businesses.length; i++){
                        if(businesses[i].status.lastLogin.date.getTime()
> aMonthLater.getTime()){
                                count++;
                        }
                }

                return count;
        },
        totalBusinesses: function(){
                return Meteor.users.find({roles: "business"}).count();
        },
        monthlySiteVisits: function(){
                date = new Date();
                return Visits.findOne({
                        month: date.getMonth(),
                        year: date.getFullYear()
                }).count;
        },
        ministerBusinesses: function(){
                usersByMinister  = Meteor.users.find({
                        roles: "business",
                        creator: Meteor.userId()
                }).fetch();

                users = _.pluck(usersByMinister, '_id');

                businesses = Business.find({
```

```
                        author: {
                                $in: users
                        }
                }).fetch();
                return businesses;
        }
});

Template.ministerBusinessesTr.helpers({
        lastLogin: function(){
                return Meteor.users.findOne({
                        _id: this.author
                }).status.lastLogin.date;
        },
        monthlyVisits: function(){
                visitsArray = this.stats_visits;
                date = new Date();
                for(var i = 0; i < visitsArray.length; i++){
                        if(visitsArray[i].month == date.getMonth() &&
visitsArray[i].year == date.getFullYear()){
                                return visitsArray[i].count;
                        }
                }
        },
        status: function(){
                aMonthLater = new Date();
                aMonthLater.setMonth(aMonthLater.getMonth() - 1);
                user = Meteor.users.findOne({
                        _id: this.author
                });

                if(user.status.lastLogin.date.getTime() <
aMonthLater.getTime())
                        return "Inactivo";
                else
                        return "Activo";
        }
});
```

- **Favorites.html**

```
<template name="Favorites">
        <div class="container">
                <table class="table table-hover">
                        <thead>
                                <tr>
                                        <th>Nombre</th>
                                        <th>Notificaciones</th>
                                        <th></th>
                                </tr>
                        </thead>
                        <tbody>
                                {{#each UserFavorites}}
                                        <tr>
                                                <td>{{this.name}}</td>
                                                <td>
                                                        {{#if
this.notifications}}

        <button class="fav-toggle btn btn-success"><i class="fa fa-toggle-
on"></i></button>
                                                        {{else}}
```

```
        <button class="fav-toggle btn"><i class="fa fa-toggle-
    off"></i></button>
                                                                {{/if}}
                                                        </td>
                                                        <td><a
    href="/business/{{this.id}}" class="btn btn-primary">Mas
    Información</a></td>
                                                    </tr>
                                                {{/each}}
                                            </tbody>
                                    </table>
        </div>
    </template>
```

- **Favorites.js**

```
    Template.Favorites.helpers({
        UserFavorites: function(){
                return Meteor.user() && Meteor.user().favorites;
        }
    });

    Template.Favorites.events({
        'click .fav-toggle':function(event){
                Meteor.call('favToggle', Meteor.userId(), this.id,
    this.notifications);
        }
    });
```

- **Profile.html**

```
    <template name="Profile">
        <div class="container">
                <h1>Perfil</h1>
                <!-- nav tabs -->
                <ul class="nav nav-tabs">
                        <li class="active"><a href="#account" aria-
    controls="account" role="tab" data-toggle="tab">Datos de
    Cuenta</a></li>
                        {{#if isInRole 'business'}}
                        <li><a href="#business" aria-controls="business"
    role="tab" data-toggle="tab">Datos de Negocio</a></li>
                        <li><a href="#promotions" aria-
    controls="promotions" role="tab" data-toggle="tab">Promociones</a></li>
                        {{/if}}
                </ul>
                <!-- Tab panes-->
                <div class="tab-content">
                        <div role="tabpanel" class="tab-pane active"
    id="account">
                                <table class="table">
                                        <tbody>
                                                <tr>

        <td><strong>Email: </strong></td>

        <td>{{EmailorName}}</td>
                                                </tr>
                                                <tr>

        <td><strong>Tipo de Cuenta: </strong></td>
```

```
    <td>{{currentUser.roles}}</td>
                                                    </tr>
                                                </tbody>
                                            </table>
                                            <a href="/admin/changepassword"
class="btn btn-primary"><i class="fa fa-pencil"></i> Cambiar
Contraseña</a>
                                            <a href="/admin/changeemail" class="btn
btn-primary"><i class="fa fa-pencil"></i> Cambiar Email</a>
                                        </div>
                                        {{#if isInRole 'business'}}
                                        <div role="tabpanel" class="tab-pane"
id="business">
                                            <br>
                                            {{#if
currentUser.profile.full_register}}
                                                <a href="/admin/businessupdate"
class="btn btn-primary"><i class="fa fa-pencil"></i> Editar Datos de
Negocio</a>
                                            {{else}}
                                                <a href="/addbusiness"
class="btn btn-primary"><i class="fa fa-plus"></i> Ingresa Datos de
Negocio</a>
                                            {{/if}}
                                            <br>
                                            <br>
                                            <table class="table table-striped">
                                                <tbody>
                                                    <tr>

    <th>Imagen/es:</th>
                                                        <td><img
src="{{images.url}}"></td>
                                                    </tr>
                                                    <tr>

    <th>Nombre:</th>

    <td>{{business.name}}</td>
                                                    </tr>
                                                    <tr>

    <th>Subtitulo:</th>

    <td>{{business.subtitle}}</td>
                                                    </tr>
                                                    <tr>

    <th>Descripción:</th>

    <td>{{business.desc}}</td>
                                                    </tr>
                                                    <tr>

    <th>Descuento:</th>

    <td>{{business.discount}}%</td>
                                                    </tr>
                                                    <tr>
```

```
        <th>Dirección:</th>

        <td>{{business.address}}</td>
                                                    </tr>
                                                    <tr>

        <th>Pais:</th>

        <td>{{business.country}}</td>
                                                    </tr>
                                                    <tr>

        <th>Ciudad:</th>

        <td>{{business.city}}</td>
                                                    </tr>
                                                    <tr>

        <th>Teléfono:</th>

        <td>{{business.phone}}</td>
                                                    </tr>
                                                    <tr>
                                                            <th>Pagina
    Web:</th>

        <td>{{business.website}}</td>
                                                    </tr>
                                                    <tr>
                                                            <th>Horario
    de atencion:</th>

        <td>{{business.atention_start}} - {{business.atention_end}}</td>
                                                    </tr>
                                                    <tr>

        <th>Categoria/as:</th>

        <td>{{business.category}}</td>
                                                    </tr>
                                                    <tr>

        <th>Productos:</th>
                                                            <td>

        {{#each business.products}}

        <li>{{this}}</li>

        {{/each}}
                                                            </td>
                                                    </tr>
                                                    <tr>

        <th>Sucursales:</th>
                                                            <td>

        {{#each business.branches}}

        <li>{{this}}</li>
```

```
        {{/each}}
                                                                    </td>
                                                                </tr>
                                                            </tbody>
                                                        </table>
                                                    </div>
                                                    <div role="tabpanel" class="tab-pane"
    id="promotions">
                                                        <br>
                                                        <a href="/admin/addpromotion" class="btn
    btn-primary btn-circle btn-lg"><i class="fa fa-plus"></i></a>
                                                        <br>
                                                        <br>
                                                        {{#each promotions}}
                                                                {{> Promotion}}
                                                        {{/each}}
                                                    </div>
                                                    {{/if}}
                                                </div>
        </div>

    </template>
```

- **Profile.js**

```
    Template.Profile.helpers({
        EmailorName: function(){
                if(Meteor.user()){
                        if(Meteor.user().profile){
                                if(Meteor.user().profile.name){
                                        return
    Meteor.user().profile.name;
                                }else
                                        return
    Meteor.user().emails[0].address;
                                }
                                else{
                                        return Meteor.user().emails[0].address;

                                }
                }
        },
        business: function(){
                return Business.findOne({author: Meteor.userId()});
        },
        images: function(){
                if(Business.findOne({author: Meteor.userId()}))
                        return
    Images.findOne(Template.Profile.__helpers.get('business').call().images
    [0]);
        },
        promotions: function(){
                promotions = Promotions.find({author: Meteor.userId()});
                return promotions;
        }
    });
```

- **PromotionUpdate.html**

```
<template name="promotionUpdate">
    {{> quickForm
            collection="Promotions"
            id="updatePromotionForm"
            type="update"
            doc=promotion
    }}
</template>
```

- **PromotionUpdate.js**

```
Template.promotionUpdate.helpers({
    promotion: function(){
            return Promotions.findOne(Session.get('promotionId'));
    }
});
```

- **Support.html**

```
<template name="Support">
    <div class="container">
            <h1>Soporte</h1>
            <form class="form-horizontal support-form col-sm-8 col-sm-
offset-2">
                    <div class="form-group">
                            <label for="typeSelect" class="control-
label"><strong>¿Que problema está teniendo?</strong></label>
                            <select id="typeSelect" class="form-
control">
                                    <option value="bugs">He
encontrado algo que no funciona bien en la página</option>
                                    <option value="page">Tengo una
consulta sobre la página</option>
                                    {{#if isInRole 'business'}}
                                            <option
value="business">Tengo un problema en mi negocio</option>
                                    {{/if}}
                            </select>
                    </div>
                    <div class="form-group">
                            <label for="subject" class="control-
label"><strong>Asunto:</strong></label>
                            <input type="text" id="subject"
name="subject" maxlength="30" class="form-control">
                    </div>
                    <div class="form-group">
                            <label for="message" class="control-
label"><strong>Mensaje:</strong></label>
                            <textarea id="message" rows="5"
name="message" class="form-control"></textarea>
                    </div>
                    <div class="form-group">
                            <button type="submit"
class="btn btn-primary">Enviar</button>
                    </div>
            </form>
    </div>
</template>
```

- **Support.js**

```
Template.Support.events({
    'submit form': function(event){
                event.preventDefault();
                var type = $('#typeSelect').val();
                var subject = $('#subject').val();
                var message = $('#message').val();

                Tickets.insert({type: type, subject: subject, messages:
[{text: message}]}, function(error){
                        if(error){
                                Bert.alert("Ha ocurrido un error,
intentelo de nuevo.", 'danger', 'growl-top-right');
                        }else{
                                Bert.alert("Su mensaje ha sido enviado
con exito y sera revisado prontamente.", 'success', 'growl-top-right');
                                $('#subject').val('');
                                $('#message').val('');
                        }
                });
    }
});

Template.Support.onRendered(function(){
    $('.support-form').validate({
                rules: {
                        subject: {
                                required: true,
                                minlength: 10
                        },
                        message: {
                                required: true,
                                minlength: 15
                        },
                },
                messages: {
                        subject: {
                                required: "Debe incluir un asunto al
mensaje.",
                                minlength: "El asunto debe tener por lo
menos {0} caracteres para mejorar la comprención del problema."
                        },
                        message: {
                                required: "Un mensaje debe ser
incluido.",
                                minlength: "Su contraseña debe tener al
menos {0} caracteres"
                        },
                }
    });
});
```

- **TicketSingle.html**

```
<template name="TicketSingle">
    <h1>{{ticket.subject}}</h1>

    <div class="messages">
            {{#each ticket.messages}}
                    <div class="message">
                            <strong>{{this.author}}</strong>
                            <p>{{this.text}}</p>
                    </div>
            {{/each}}
    </div>

    <form>
            <div class="input-group">
                    <input type="text" class="form-control"
id="response" placeholder="Escriba su mensaje">
                    <span class="input-group-btn"><button class="btn
btn-primary">Responder</button></span>
            </div>
    </form>
</template>
```

- **TicketSingle.js**

```
Template.TicketSingle.helpers({
    ticket: function(){
            return Tickets.findOne(FlowRouter.getParam("id"));
    }
});

Template.TicketSingle.onRendered(function(){
    if(Roles.userIsInRole(Meteor.userId(), ['normal-user',
'business'])){
            console.log("aqui");
    }
});

Template.TicketSingle.events({
    'submit form': function(event) {
            event.preventDefault();
            var text = event.target.response.value;
            Tickets.update({_id: FlowRouter.getParam("id")} , {
                    $push: {
                            messages: {
                                    text: text
                            }
                    }
            });
            event.target.response.value = "";
    }
});
```

- **Users.html**

```
<template name="Users">
    {{#if isInRole 'super-admin, minister'}}
            <h1>Usuarios</h1>
            <a href="/createaccount" class="btn btn-primary btn-circle
btn-lg" data-toggle="tooltip" data-placement="right" title="Crear una
nueva cuenta"><i class="glyphicon glyphicon-plus"></i></a>
            <table class="table table-hover">
                    <thead>
                    <tr>
                            <th>ID</th>
                            <th>Nombre/Email</th>
                            <th>Roles</th>
                    </tr>
                    </thead>
                    <tbody>
                            {{#each users}}
                    <tr>
                                    <td>{{_id}}</td>
                                    <td>{{EmailorName}}</td>
                                    <td>{{roles}}</td>
                    </tr>
                            {{/each}}
                    </tbody>
            </table>
    {{else}}
            {{> unauthorized}}
    {{/if}}
</template>
```

- **Users.js**

```
Template.Users.helpers({
    users: function(){
            if (Roles.userIsInRole(Meteor.userId(), 'super-admin')) {
                    return Meteor.users.find({});
            }else{
                    return Meteor.users.find({creator:
Meteor.userId()});
            }
    },
    EmailorName: function(){
            if(this.profile && this.profile.name){
                    return this.profile.name;
            }else{
                    if(this.emails){
                            return this.emails[0].address;
                    }
            }
    }
});

Template.Users.onRendered(function(){
    $('[data-toggle="tooltip"]').tooltip();
});
```

- **addPromotions.html**

```
<template name="addPromotion">
    <div class="container">
            <h1>Agregar Nueva Promocion</h1>
            {{> quickForm
                    collection="Promotions"
                    id="addPromotion"
                    type="insert"
            }}
    </div>
</template>
```

- **adminBusinesses**

```
<template name="adminBusinesses">
    {{#if isInRole 'super-admin, minister'}}
            <h1>Administrar Negocios</h1>
            <table class="table table-hover">
                    <thead>
                    <tr>
                            <th>ID</th>
                            <th>Nombre</th>
                            <th>Dirección</th>
                            <th>Catalogo</th>
                            <th>Ranking en Campaña</th>
                    </tr>
                    </thead>
                    <tbody>
                            {{#each businesses}}
                                    {{> businessTr}}
                            {{/each}}
                    </tbody>
            </table>
    {{else}}
            {{> unauthorized}}
    {{/if}}
</template>

<template name="businessTr">
    <tr>
            <td>{{_id}}</td>
            <td>{{name}}</td>
            <td>{{address}}</td>
            <td><select class="form-control denomination">
                    <option value="Amigo">Amigo</option>
                    <option value="Ministerio">Ministerio</option>
            </select></td>
            <td><select class="form-control rank">
                    <option value="1">1 Cuchara</option>
                    <option value="2">2 Cucharas</option>
                    <option value="3">3 Cucharas</option>
                    <option value="4">4 Cucharas</option>
                    <option value="5">5 Cucharas</option>
            </select></td>
            <td><button class="btn btn-primary
savebtn">Guardar</button></td>
    </tr>
</template>
```

- **adminBusinesses.js**

```
Template.adminBusinesses.helpers({
    businesses: function(){
            if(Roles.userIsInRole(Meteor.userId(), 'super-admin')){
                    return Business.find({});
            }
            //si es Ministro entonces
            var usersId = _.pluck(Meteor.users.find({creator:
Meteor.userId()}).fetch(), "_id");
            var businesses = Business.find({
                                    author: {
                                            $in: usersId
                                    }
                            });
            return businesses;
    }
});

Template.businessTr.onRendered(function(){
    $('select.denomination').val(this.data.denomination);
    $('select.rank').val(this.data.campaign_rank);
});
```

- **adminTickets.html**

```
<template name="adminTickets">
    <h1>Tickets de Soporte</h1>
    <div class="pull-right">
            <div class="btn-group">
                    <button type="button" class="btn btn-success btn-
filter" data-target="business">Negocio</button>
                    <button type="button" class="btn btn-warning btn-
filter" data-target="page">Página</button>
                    <button type="button" class="btn btn-danger btn-
filter" data-target="bugs">Errores</button>
                    <button type="button" class="btn btn-default btn-
filter" data-target="all">Todos</button>
            </div>
    </div>
    <table class="table table-hover">
            <thead>
                    <tr>
                            <th>Asunto</th>
                            <th>Fecha de Creación</th>
                            <th>Tipo</th>
                            <th></th>
                    </tr>
            </thead>
            <tbody>
                    {{#each tickets}}
                            {{> ticketTr}}
                    {{/each}}
            </tbody>
    </table>
</template>

<template name="ticketTr">
    <tr data-status={{type}} data-notread={{Notread}}>
            <td>{{subject}}</td>
            <td>{{moment}}</td>
            <td>{{tipo}}</td>
```

```
                    <td><a href="/ticket/{{_id}}" class="btn btn-
        default">Responder</a></td>
                </tr>
        </template>
```

**adminTickets.js**

```
Template.adminTickets.helpers({
        tickets: function(){
                if(Roles.userIsInRole(Meteor.userId(), 'normal-user')){
                        return Tickets.find({authorId: Meteor.userId()},
{sort: {read: 1}});
                }

                if(Roles.userIsInRole(Meteor.userId(), 'super-admin')){
                        return Tickets.find({type: "bugs"}, {sort: {read:
1}});
                }
        }
});

Template.adminTickets.onRendered(function(){
        $('.btn-filter').on('click', function () {
                var $target = $(this).data('target');
                if ($target != 'all') {
                        $('.table tbody tr').css('display', 'none');
                        $('.table tr[data-status="' + $target +
'"]').fadeIn('slow');
                } else {
                        $('.table tbody tr').css('display',
'none').fadeIn('slow');
                }
        });
});

Template.ticketTr.helpers({
        Notread: function(){
                if(Roles.userIsInRole(Meteor.userId(), 'normal-user')){
                        return !this.author_read;
                }else
                        return !this.support_read;
        },
        moment: function(){
                moment = moment(this.createdAt);
                moment.locale('es');
                return moment.fromNow() +" ("+ moment.format('DD/MMM/YY') +
(")");
        },
        tipo: function(){
                switch(this.type){
                        case "bugs":
                                return "Errores";
                                break;
                        case "page":
                                return "Página";
                                break;
                        case "business":
                                return "Negocio";
                                break;
                }
        }
});
```

- **businessTr.js**

```
Template.businessTr.events({
    'click .savebtn':function(event){
                var denomination =
event.target.parentElement.parentElement.cells[3].lastChild.value;
                var rank =
event.target.parentElement.parentElement.cells[4].lastChild.value;

                Business.update(this._id,{
                        $set: {
                                denomination: denomination,
                                campaign_rank: rank
                        }
                }, function(error){
                        if(error){
                                Bert.alert({
                                        message: error,
                                        type: 'danger',
                                        style: 'growl-top-right'
                                });
                        }else{
                                Bert.alert({
                                        message: 'Datos almacenados con
éxito',
                                        type: 'success',
                                        style: 'growl-top-right'
                                });
                        }
                });
    }
});
```

- **changeEmail.html**

```
<template name="changeEmail">
    <div class="container">
                <h2 class="col-sm-offset-4">Cambiar Email</h2>
                <form class="changePass form-horizontal form-signin">
                        <div class="form-group">
                                <label for="newEmail" class="col-sm-5
control-label" >Nuevo Email:</label>
                                <div class="col-sm-7">
                                        <input type="email"
class="form-control" id="newEmail" name="newEmail" placeholder="Nueva
dirección">
                                </div>
                        </div>

                        <div class="form-group">
                                <div class="col-sm-offset-5
col-sm-7">
                                        <button type="submit"
class="btn btn-primary">Guardar</button>
                                </div>
                        </div>
                </form>
    </div>
</template>
```

- **changeEmail.js**

```
Template.changeEmail.events({
    'submit form': function(event) {
                        event.preventDefault();
                        var newEmail = event.target.newEmail.value;

                        if(Meteor.user().emails){
                        if(newEmail === Meteor.user().emails[0].address){
                                Bert.alert("El nuevo email debe ser
diferente al anterior.", 'warning', 'growl-top-right');
                        }else{
                                Meteor.call('changeEmail',
Meteor.userId(), Meteor.user().emails[0].address, newEmail);
                                event.target.newEmail.value = '';
                                Bert.alert("Su Email ha sido cambiado
exitosamente", 'success', 'growl-top-right');
                        }
                        }else{
                                Meteor.call('changeEmail',
Meteor.userId(), "", newEmail);
                                event.target.newEmail.value = '';
                                Bert.alert("Su Email ha sido cambiado
exitosamente", 'success', 'growl-top-right');
                        }
                }
});

Template.changeEmail.onRendered(function(){
    $('.changePass').validate({
                rules: {
                        newEmail: {
                                required: true,
                                email: true
                        },
                },
                messages: {
                        newEmail: {
                                required: "Debe ingresar alguna
dirección.",
                                email: "Debe incluir un @ en la
dirección de correo electrónico."
                        },
                }
    });
});
```

- **changePassword.html**

```
<template name="changePassword">
    <div class="container">
            <h2 class="col-sm-offset-4">Cambiar Contraseña</h2>
            <form class="changePass form-horizontal form-signin">
                    <div class="form-group">
                            <label for="oldPassword" class="col-sm-4
control-label" >Contraseña actual:</label>
                            <div class="col-sm-8">
                                    <input type="password"
class="form-control" id="oldPassword" name="oldPassword"
placeholder="Contraseña actual">
                            </div>
                    </div>
                    <div class="form-group">
                            <label for="newPassword" class="col-sm-4
control-label" >Contraseña nueva:</label>
                            <div class="col-sm-8">
                                    <input type="password"
class="form-control" id="newPassword" name="newPassword"
placeholder="Contraseña nueva">
                            </div>
                    </div>
                    <div class="form-group">
                            <div class="col-sm-offset-4
col-sm-8">
                                    <button type="submit"
class="btn btn-primary">Guardar</button>
                            </div>
                    </div>
            </form>
    </div>
</template>
```

- **changePassword.js**

```
Template.changePassword.events({
    'submit form': function(event) {
                    event.preventDefault();
                    var oldPassword = event.target.oldPassword.value;
                    var newPassword = event.target.newPassword.value;

                    if(oldPassword === newPassword){
                            Bert.alert("Las contraseñas deben ser
diferentes", 'danger', 'growl-top-right');
                    }else{
                            Accounts.changePassword(oldPassword,
newPassword, function(error){
                                    if(error){

    Bert.alert(error.reason, 'danger', 'growl-top-right');
                                    }else{
                                            Bert.alert("La
contraseña ha sido cambiada exitosamente", 'success', 'growl-top-
right');

    event.target.oldPassword.value = '';
    event.target.newPassword.value = '';
                                    }
                            });
                    }
```

```
                }
});

Template.changePassword.onRendered(function(){
    $('.changePass').validate({
                rules: {
                        oldPassword: {
                                required: true,
                                minlength: 6
                        },
                        newPassword: {
                                required: true,
                                minlength: 6
                        },
                },
                messages: {
                        oldPassword: {
                                required: "Una contraseña es
obligatoria",
                                minlength: "Su contraseña debe tener al
menos {0} caracteres"
                        },
                        newPassword: {
                                required: "Una contraseña es
obligatoria",
                                minlength: "Su contraseña debe tener al
menos {0} caracteres"
                        },
                }
    });
});
```

# 7 FACTIBILIDAD

## 7.1 Introducción

En este capítulo se determinará si la implementación de la solución propuesta es factible tanto en términos técnicos como económicos.

## 7.2 Factibilidad Técnica

Como pudimos comprobar, MongoDB es totalmente capaz de almacenar y manejas los datos requeridos y proporcionar las características necesarias para implementar la solución. Al no ser necesario un tratamiento más exhaustivo de la información MongoDB es la mejor opción al mismo tiempo que promete velocidad en el despliegue de los datos almacenados.

En cuanto a la infraestructura necesaria, La Revolución de la Cuchara ya posee los servidores necesarios siendo requerida solamente la instalación de software específico.

Referente a los recursos humanos, La Revolución de la Cuchara cuenta con gran cantidad de voluntarios con conocimientos informáticos además del autor de proyecto de software por lo que la implementación de la solución es totalmente factible. Además, las herramientas utilizadas cuentan con vasta documentación que cualquiera puede revisar para comprender el software diseñado y mejorarlo o modificarlo en el futuro.

## 7.3 Factibilidad Económica

### 7.3.1 Recursos humanos

Al tratarse de un voluntariado, el trabajo realizado por el personal está libre de costos relacionados recursos humanos. Aun así, se ha realizado una estimación de costos como referencia. Vale decir que para poder estimar el tiempo requerido para desarrollar la aplicación se han analizado diferentes opiniones de expertos en los foros del framework Meteor. Esto se debe a que el desarrollo en dicho framework difiere en gran medida al tipo de desarrollo en que se ubica una estimación basada en “Puntos de casos de uso”.

La estimación que se presenta a continuación se realizó considerando la experiencia del ingeniero que varían de Junior, 0 a 1 año de experiencia, y Senior, con 3 a 5 años de experiencia.

| Ingeniero | Tiempo de desarrollo | Costo Mensual | Total |
|---|---|---|---|
| Junior | 3 meses | $867.405 | $ 2.602.215 |
| Senior | 1 mes | $1.303.313 | $1.303.313 |

*Tabla 3 Estimacion de costos de recursos humanos.Licencias de software*

[Fuente: http://conexioningenieros.com/wp-content/uploads/2016/11/Estudio-de-Sueldos-Conexión-Ingenieros-2016.pdf]

### 7.3.2 Licencias de software

Todas las herramientas utilizadas para el desarrollo e implementación de la solución entran en la categoría de software libre o gratuito por lo que no es incurre en gastos por adquisición de licencias.

El sistema operativo necesario es Linux por lo que tampoco se incurre en inversión alguna.

Como se menciona anteriormente, La revolución de la cuchara cuenta con los servidores necesarios para la implementación.

### 7.3.3 Conclusión

Como ya se ha visto, el proyecto es factible técnica y económicamente gracias a la gratuidad de las licencias y de los trabajos voluntarios.

# 8 PRUEBAS

## 8.1 Introducción

En este capítulo se describirán los tipos de pruebas realizadas para asegurar la calidad del producto final. Se incluirá también una plantilla de prueba de usabilidad para ser realizada con los actores finales del sistema.

## 8.2 Pruebas de Caja Negra

Una prueba de caja negra busca encontrar errores en las entradas y salidas de un software sin tener en cuenta la forma en la que estos tratan la información internamente.

Este tipo de prueba fue realizada en cada sprint incremental del software desarrollado siendo depuradas todas las falencias encontradas.

Esto asegura que cada sección funciona de la manera predicha, aunque no asegura que todos los errores hayan sido encontrados.

Como todo desarrollo de software utilizando la metodología ágil, la retroalimentación de los usuarios es vita y se ha proporcionado un apartado para ello dentro de la plataforma.

## 8.3 Pruebas de Usabilidad

### 8.3.1 Que es una prueba de usabilidad

Las pruebas de usabilidad buscan entender como el usuario se relaciona con un software específico y ayudar en el diseño de las interfaces en medro de la experiencia de usuario (UX).

Estas pruebas darán como resultado pantallas auto explicativas, intuitivas y fáciles de navegar permitiendo mejorar el desempeño de sus actores en las diversas tareas que realizarán en la plataforma.

Esta prueba de usabilidad se enfoca en los siguientes ámbitos:

- Identidad
- Contenido
- Navegación
- Grafica Web
- Búsqueda
- Retroalimentación
- Utilidad

### 8.3.2 Preguntas sobre Identidad

Estas preguntas buscan definir si el usuario es capaz de reconocer a primera vista en que sitio a entrado y como logra éste diferenciarse de otros sitios. Responde a la inquietud de si se logra transmitir al navegante la imagen que queremos tener como organización.

Estas preguntas se deben realizar antes de que el usuario haga el primer clic.

### 8.3.3 Preguntas sobre Contenido

Estas preguntas se realizarán luego de permitir al usuario interactuar con la plataforma.

El objetivo de estas es clarificar la calidad de la presentación informativa. El cómo se visualiza el contenido y lo comprensible que este es a lo largo de la aplicación.

### 8.3.4 Preguntas sobre Navegación

Las preguntas sobre navegación permiten saber concretamente si la forma de organizar la información es idónea de acuerdo a la experiencia, conocimientos y expectativas que tenga el usuario.

Estas preguntas deben ser realizadas mientras el usuario navega por la aplicación. Es importante que la persona que ejecuta esta prueba note si el usuario hace uso de los elementos de navegación de la aplicación por sobre los botones del navegador que este usando.

### 8.3.5 Preguntas sobre Gráfica Web

Estas preguntas dejan en manifiesto si la información gráfica le ha sido de ayuda para el usuario. También denota su percepción sobre la velocidad de despliegue de la información.

### 8.3.6 Preguntas sobre Búsqueda

Estas preguntas responden la incógnita de si las formas de búsqueda han sido propicias para conseguir la información que el usuario necesita.

### 8.3.7 Preguntas sobre Retroalimentación

Las preguntas sobre retroalimentación responden si el usuario encuentra fácilmente la forma de ponerse en contacto con la organización.

### 8.3.8 Preguntas sobre Utilidad

Estas preguntas son utilizadas para determinar un resumen general de la experiencia de usuario.

### 8.3.9 Como analizar los resultados

La prueba de usabilidad permitirá comprender cuales son las tareas más difíciles de completar para los usuarios e identificar los elementos menos comprensibles. Estos datos deberán evaluarse por el equipo de desarrollo para mejorar la experiencia de los usuarios y disminuir la frustración al momento de utilizar la aplicación.

Como estadística referencial podemos sumar los resultados de todas las preguntas y el resultado representara el grado de satisfacción conseguida por el usuario. Mientras más bajo sea este valor menor es el éxito de la interfaz de usuario actual y requerirá de una revisión del equipo.

## PRUEBA DE USABILIDAD

**Entrevistador:**
**Usuario:**
**Fecha:**

Marque con una X su respuesta. Las respuestas se ordenan de lo más negativo a lo más positivo.

**Identidad**

| Contenido | Muy Negativo | Normal | | | Positivo |
|---|---|---|---|---|---|
| **1.** ¿Ha podido identificar a que organización pertenece este sitio fácilmente? | 1 | 2 | 3 | 4 | 5 |
| **2.** ¿ Ha comprendido a que campaña brinda apoyo esta plataforma fácilmente? | 1 | 2 | 3 | 4 | 5 |
| **3.** ¿Ha encontrado una forma de contactar a la organización? ¿Fue fácil encontrarla? | 1 | 2 | 3 | 4 | 5 |
| **4.** ¿Distingue alguna imagen que represente a la institución? ¿Está en una buena posición en la pagina? | 1 | 2 | 3 | 4 | 5 |
| **5.** ¿Existe algún elemento que usted encuentra fuera de lugar o que no se relaciona con la organización o la campaña? Cual: | 1 | 2 | 3 | 4 | 5 |
| **6.** ¿Se entiende correctamente hacia que audiencia está dirigida está plataforma? Mencione la audiencia: | 1 | 2 | 3 | 4 | 5 |

**Contenido**

| Contenido | Muy Negativo | Normal | | | Positivo |
|---|---|---|---|---|---|
| **1.** ¿Le ha resultado fácil reconocer el área más importante de la plataforma? | 1 | 2 | 3 | 4 | 5 |
| **2.** ¿Le pareció adecuada la información presente en la pantalla principal? | 1 | 2 | 3 | 4 | 5 |
| **3.** ¿Los textos incluidos en links y botones son suficientemente descriptivos para entender lo que hacen o hacia dónde dirigen? | 1 | 2 | 3 | 4 | 5 |
| **4.** ¿Considera que los elementos en cada entrada de negocio facilitan la identificación de los datos importantes? ¿Contiene todos los datos necesarios? | 1 | 2 | 3 | 4 | 5 |

**Navegación**

| Contenido | Muy Negativo | Normal | | | Positivo |
|---|---|---|---|---|---|
| **1.** ¿Le ha resultado fácil reconocer el área más importante de la plataforma? | 1 | 2 | 3 | 4 | 5 |
| **2.** ¿Puede comprender de qué manera se navega por el sitio? ¿Se distingue fácilmente? | 1 | 2 | 3 | 4 | 5 |
| **3.** ¿Con los elementos presentes en las páginas puede saber dónde se encuentra y como volver atrás sin usar los botones del navegador? | 1 | 2 | 3 | 4 | 5 |

**Grafica Web**

| Contenido | Muy Negativo | Normal | | | Positivo |
|---|---|---|---|---|---|
| **1.** ¿Le ha parecido adecuada la forma de presentar las imágenes en el sitio? ¿Presentan de manera satisfactoria el contenido del que trata la plataforma? | 1 | 2 | 3 | 4 | 5 |
| **2.** ¿En algún momento sucedió que tuvo que seguir navegando sin que estas llegaran a cargar totalmente? ¿Es el sitio suficientemente rápido? | 1 | 2 | 3 | 4 | 5 |
| **3.** ¿Considera que el sitio es equilibrado gráficamente? ¿Lo ha encontrado muy recargado? | 1 | 2 | 3 | 4 | 5 |
| **4.** ¿Han llamado su atención las imágenes presentadas para llegar a hacer clic sobre ellas? | | | | | |

**Búsqueda**

| Contenido | Muy Negativo | Normal | | | Positivo |
|---|---|---|---|---|---|
| **1.** ¿Pudo reconocer a simple vista si la plataforma ofrece un buscador? | 1 | 2 | 3 | 4 | 5 |
| **2.** ¿Fueron los resultados de sus búsquedas acertados con lo que esperaba encontrar? | 1 | 2 | 3 | 4 | 5 |
| **3.** ¿La cantidad de resultados por página fueron suficientes? | 1 | 2 | 3 | 4 | 5 |

**Retroalimentación**

| Contenido | Muy Negativo | Normal | | | Positivo |
|---|---|---|---|---|---|
| **1.** ¿Encuentra una forma de realizar consultas, sugerencias o comentarios vía la misma plataforma? ¿Fue fácil encontrarla? | 1 | 2 | 3 | 4 | 5 |
| **2.** ¿Le ha resultado útil la forma en la que la plataforma muestra si su consulta ha sido revisada? | 1 | 2 | 3 | 4 | 5 |

**Utilidad**

| Contenido | Muy Negativo | Normal | | | Positivo |
|---|---|---|---|---|---|
| **1.** ¿Luego de la primera mirada pudo entender cuál es el objetivo de la plataforma? | 1 | 2 | 3 | 4 | 5 |
| **2.** ¿El contenido que ofrece la plataforma son de utilidad para su caso personal? | 1 | 2 | 3 | 4 | 5 |

1. ¿Qué fue lo que más le llamo la atención positiva o negativamente de la plataforma?

2. ¿Tiene algún comentario que no haya sido abordado en alguna pregunta anterior?

# 9 CONCLUSIÓN

La tecnología tiene la cualidad de evolucionar rápidamente, amoldándose a las nuevas necesidades del mundo, desarrollando nuevos paradigmas y formas de solucionar problemas aprovechando los recursos de los que disponemos. Uno de estos recursos es el espacio de almacenamiento, el cual se ha visto reducido en costos en el último tiempo, y las tecnologías NoSQL han probado hacer un excelente trabajo explotándolo. Uno de los ejemplos que más destaca dentro de estas tecnologías es MongoDB por su avanzado estado de desarrollo, tamaño de su comunidad y documentación exhaustiva. Por otro lado, su desarrollo es OpenSource que implica ser una alternativa de costo cero lo que la convierte en una cualidad deseable para cualquier organización sin fines de lucro.

Por otro lado, una de los aspectos más importantes para la campaña es la experiencia del usuario. Para aquello se estudió qué nos ofrece Meteor y su gama de productos paralelos para mejorar dicho aspecto. Este estudio nos mostró la versatilidad del framework, su facilidad de compresión gracias a la capacidad de ordenar códigos por plantillas y nos presenta conceptos como sitios de una sola página, o lo que es lo mismo, la carga dinámica de contenido en una sola página html por medio de javascript bajo el framework Blaze.

Pero la pregunta de cómo satisfacer las necesidades del cliente en términos de comodidad y experiencia siempre ha sido muy difícil de responder dada las variadas formas de relacionarse con los sistemas. Esto lo podemos resolver aplicando una prueba de usabilidad y es justamente lo que realizamos al final de este proyecto de software. Dichas pruebas no pueden demostrar su utilidad a priori por lo que son pensadas para ser aplicadas iterativamente y guiar nuevos cambios a la plataforma. De esta forma la plataforma podrá responder a las nuevas maneras de interactuar con las interfaces como lo hemos visto en años anteriores con ejemplos como la interfaz de Windows y los dispositivos touch.

La utilización de estas tecnologías nos ha enseñado que existen una enorme cantidad de opciones a la hora de presentar una solución a los clientes. Con los diferentes requerimientos que pueden presentarnos, ahora podemos comprender mejor que la misma estrategia no siempre es aplicable. Es decir, podemos obviar ciertas características de SQL por no ser necesarias para nuestro producto y aplicar una alternativa que nos entregue nuevas características no presentes, incompletas o no aplicables con el paradigma clásico.

Como resumen general revisaremos el cumplimiento de cada objetivo específico presentado al inicio de este documento, específicamente en el capítulo 2.

- Diseñar una nueva interfaz web para el sitio “The Golden Pages” bajo criterios de:
    - Usabilidad
    - Tiempo de Respuesta
    - Navegabilidad

Pudimos comprobar el criterio de usabilidad y navegabilidad a través del uso de pruebas de usabilidad las que no solo son aplicadas en el desarrollo del software, sino que iterativamente. Este criterio ha sido validado en general para nuestros clientes más no para cada cliente.

El tiempo de respuesta no pudo ser verificado, pero se puede entender como cumplido ya que es precisamente una característica de Meteor y MongoDB conjuntamente.

- Diseñar plataforma de gestión para el sitio "The Golden Pages" que facilite la incorporación y gestión de servicios por parte de sus propios actores, como:
    - Promociones
    - Noticias
    - Favoritos
    - Calificaciones

Este objetivo ha sido cumplido a cabalidad como se ha demostrado con los diseños presentados en este trabajo de tesis. No nos queda más por hablar de este punto así que no moveremos al siguiente sin mucho detalle.

- Diseñar una plataforma que facilite el seguimiento de los diversos actores que participan del sitio "The Golden Pages", con herramientas como:
    - Estadísticas tanto para administradores como para cada dueño de negocio.
    - Notificaciones de errores reportados.

El criterio que nos habla sobre las estadísticas útiles ha sido cubierto hasta donde nuestro cliente ha indicado necesidad. Pero es posible generar más estadísticas en el futuro en base a los datos almacenados en la base de datos.

El segundo criterio también ha sido cubierto a través de un apartado donde los clientes pueden contactarse con los administradores y presentar sus problemas para ser revisados por el equipo. De esta forma el equipo responsable podrá obtener una retroalimentación directa de sus clientes como es costumbre en la metodología de desarrollo ágil.

- Verificar la calidad del producto final.

Este objetivo ha sido cumplido con cierta completitud ya que como se ha mencionado antes la calidad es una cualidad muy difícil de satisfacer pero que hemos verificado hasta donde nuestro cliente se ha sentido satisfecho.

Con esto podemos tener una visión general de lo que hemos revisado y cumplido con este proyecto de tesis para terminar con una meditación.

El futuro se muestra grandioso para la innovación, debemos tomar parte en él y abalanzarnos con optimismo al progreso, siempre en la búsqueda de la satisfacción de todos. Los grandes han tomado con valentía los retos de su tiempo y se han elevado por encima del conformismo. Este proyecto de software intenta no ser der uno más y visualizar el problema con otra perspectiva.

# 10 BIBLIOGRAFIA

[1] (s.f.). Quiénes somos. La Revolución de la Cuchara. Obtenido 2, 2017, de http://agilemanifesto.org/

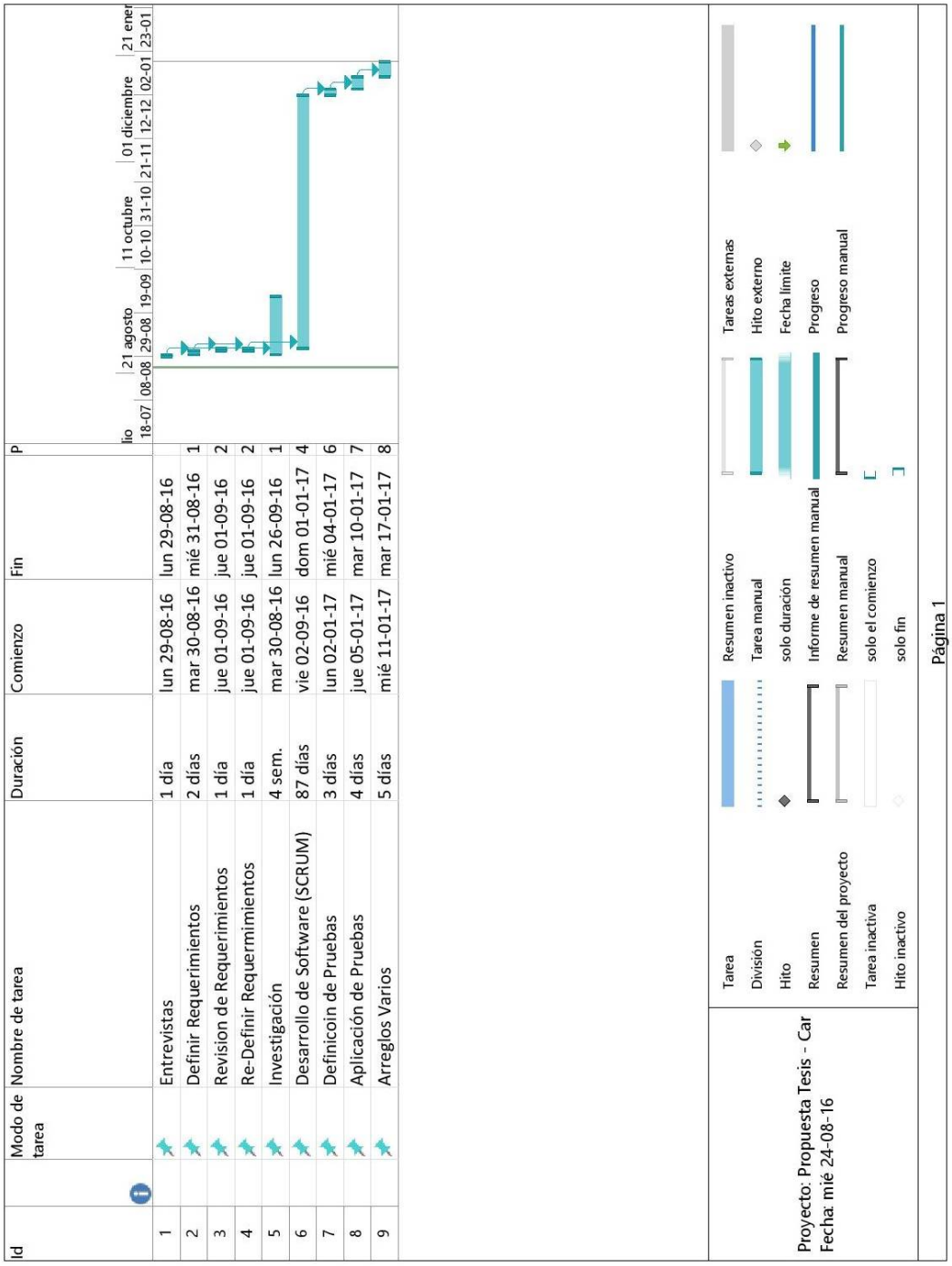
[2] (s.f.). Manifesto for Agile Software Development. Obtenido 11, 2016, de http://www.larevoluciondelacuchara.org/quienes-somos/

[3] (s.f.). Teorema CAP. Wikipedia. Obtenido 11, 2016, de https://es.wikipedia.org/wiki/Teorema_CAP

[4] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes y Robert E. Gruber. (2006, Noviembre). Bigtable: A Distributed Storage System for Structured Data. Paper presentado en OSDI'06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA.

[5] Introduction to Latency Compensation. MeteorHacks. Obtenido 11, 2016, de https://meteorhacks.com/introduction-to-latency-compensation/

# 11 ANEXO: PLANIFICACION




La etapa de desarrollo SCRUM se divide en sprints de 1 semana.