UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria
de Vilanova i la Geltrú

Desplegament de la Infraestructura i Arquitectura, l'aplicació mòbil, i la Base de Dades

Equip B2

Índex...

Descripció general Sprint

Durant aquest Sprint hem decidit avançar totes les seccions del nostre projecte (Desplegament de l'Infraestructura, Aplicació Mòbil i Base de dades). Considerem que l'equip ha treballat força bé i s'han avançant gran part del material que s'havia planificat fer de cara en aquest Sprint.

Desplegament de l'Infraestructura i Arquitectura

Descripción general

Durante este sprint hemos resuelto las dudas propuestas por el cliente respecto tanto a la implementación física como el funcionamiento de nuestra infraestructura con la API (realizada por el grupo B3).

La arquitectura sigue el modelo del sprint anterior, pero hemos profundizado en los aspectos que considerábamos más esenciales para plantear y empezar un pequeño prototipo de la implementación del sistema.

Siguiendo el modelo anterior, podemos dividir claramente la infraestructura en los sectores: cloud, fog y edge.

Cloud

En el caso del aeropuerto real, el cloud será contratado en una empresa que ofrezca sus servicios y nos interesen sus características. Por ejemplo Amazon Web Services sería una buena opción.

De cara a la implementación que realizaremos, teníamos pensado realizar la implementación final con una Raspberry Pi corriendo Raspbian (que nos pone a disposición los profesores de esta asignatura), pero debido al confinamiento, la realizaremos con una máquina virtual conectada al Fog en otra red local (creando e implementando una VPN) o en una misma red local (la red de la terminal estaría en una red interna dentro de una máquina).

El cloud se ocupará de las operaciones que requieran datos "no urgentes", como podría ser información de los vuelos, usuarios registrados en la app, información del estado de los coches, tiendas, ofertas….

Fog

En el caso del aeropuerto real, habrá una sala medianamente céntrica en el aeropuerto, la cual alojará todos los servidores necesarios para el funcionamiento del sistema. Es decir, sus 3 servidores fog mas 2 servidores backup que empezarían a funcionar en caso de caída de cualquiera de los anteriores.

De todas maneras, para tener una buena escalabilidad, la ampliacion/modificacion del numero de servidores será sencilla, permitiendo poder adaptarnos a un gran crecimiento del número de dispositivos conectados en cualquier momento.

Para realizar la simulación del FOG utilizaremos un conjunto de 5 máquinas virtuales con el SO de Debian 10.3 . Por ultimo haremos que estas máquinas virtuales tengan direcciones IP estáticas (para facilitar la implementación de los métodos de elección y la comunicación). Estas 5 IP's aparte de ser estáticas y no estar reservadas para estas máquinas (no están dentro de la pool de DHCP) se guardarán en nuestro programa.

En caso de que se caiga una MV (es decir, esa IP no estará Operativa), una de las 2 maquinas de repuesto se levantara y ejecutará el programa, primeramente preguntando a las máquinas que quedan encendidas cual es la dirección IP de la caída (y entonces se creará la interfaz estática con esta IP).

Otra opcion seria que estas direcciones IP fuesen dinámicas. En primer caso, tendrán que conocerse, pudiéndose realizar esto mediante un broadcast a toda la red preguntando quien es una máquina que pertenece a FOG en un puerto, quien responda (con IP o MAC, si sabemos uno conoceremos el otro) la pondremos dentro de una tabla que indica quien es una máquina de FOG.

En caso de que se cayese una de las máquinas FOG, la máquina backup tendría que comprobar cual de estas es la máquina que no responde (se puede conocer si está "dormida" y va preguntando a las máquinas FOG si están vivas, en el momento que una no responda a X mensajes, se puede considerar caída y entonces enviar un mensaje a todo el mundo con la opción de: cambio de máquina fog, IP Y por IP Z).

Edge

El edge se forma con el conjunto de todos los dispositivos que se podrán conectar al nuestro sistema. Por lo tanto, diferenciaremos entre cada tipo de dispositivo.

Terminal

Nos referiremos a una terminal a todo aquello que no pueda ser descrito como dispositivo móvil, sensor o coche y se conecte a la capa de edge.

Un ejemplo podría ser el ordenador que tienen los azafatas al realizar el embarque y sean capaces de ver donde se encuentran los pasajeros, los pasajeros que faltan, aquellos que no van a aparecer (por ejemplo los que llegan tarde, se indicarán como no conectados a la red). Además también contarán con la opción de enviar un coche a buscar al pasajero que se encuentre lejos (esto también lo realiza automáticamente el programa, pero siempre se tiene que ofrecer la posibilidad de realizarlo manualmente).

Por otro lado, los terminales de pasajeros serán maquinas (parecidas a una máquina de sacar billetes del metro), los cuales estarán plantados en varios sitios del aeropuerto. Estos constaran de una pantalla táctil en la que los pasajeros podrán solicitar un coche.

Como los terminales tienen una posición fija, no será necesario localizar-los, simplemente si estos envian un numero identificador, el cloud (el cual tendrá una tabla de terminal - localización) ja sabrá donde debe mandar el coche.

Las funciones de una terminal para los pasajeros serán las siguientes:

- Petición de coches
- Mostrar las ofertas cercanas más interesantes
- Mostrar el mapa de la terminal a los usuarios
- Indicar dónde se encuentra su puerta de embarque
- Indicar dónde se encuentra el baño más cercano.

Respecto a la implementación que haremos nosotros, la terminal se podrá simular con un pequeño programa de python en una máquina virtual que simplemente genere llamadas a FOG, realizando las acciones listadas anteriormente

Coche

En el caso del aeropuerto real, dispondremos de coches que irán a buscar a los usuarios del aeropuerto y los llevaran a donde ellos deseen.

Para la simulación que realizaremos nosotros, podemos realizar una aproximación con una máquina virtual la cual corre Debian y reciba/realice llamadas a la FOG. Estas llamadas serán del tipo: Estoy en X sitio, Estoy ocupado/cargando/transportando/libre Y las comandas entrantes que puede esperar pueden ser: Ves a X.

Por último en el caso de llamadas entrantes como pedir un coche simplemente serán una redirección del FOG a un dispositivo del tipo coche cambiando su estado en la BD, y una vez finalice el transporte actualizará su posición y su estado

Dispositivos móviles

El dispositivo móvil se conectaran a la infraestructura a través de la aplicación que también estamos desarrollando, por lo tanto su implementación es la propia aplicación.
La información sobre la aplicación se encuentra en su propio apartado.

Sensores

Los sensores del aeropuerto hacen referencia a los dispositivos que nos permiten localizar los elementos que nos interesan (coches, personas…).

Hemos decidido que en nuestro esta función la realizaran los routers que hemos situado por el aeropuerto según el siguiente mapa:

Los routers son los signos WI-FI repartidos por el mapa.
Estos routers se encargan de hacer ping a los respectivos dispositivos i a través de los resultados obtenidos (tiempo de respuesta) nos permitirán situar al dispositivo (coche,mobil…).

Comunicación entre los elementos de la Arquitectura y Infraestructura

Hasta ahora hemos visto los elementos del sistema y su planteamiento, a continuación vamos a ver la forma en la que estos se comunican entre sí (la cual varía según los tipos de dispositivos involucrados).

Cloud <-> Fog

Cuando el fog reciba una petición de alguno de los dispositivos que requiere escalar la operación al cloud, este enviará la solicitud al cloud. Después el cloud devolverá la información solicitada al fog i éste la enviará al dispositivo solicitante.

Por ejemplo, cuando se reciba una solicitud que requiera información de los vuelos, (su hora de salida, pista de embarque)..., usuarios registrados en la app, información de los coches tipo su estado, si estan averiados, etc. Tiendas del aeropuerto, ofertas….

Fog <-> Coche

El fog mantendrá una lista con todos los coches (su numero de identificacion, ip, localizacion…) del aeropuerto, juntamente con su estado (disponible, en carga,ocupado…).

Una vez alguien solicite un coche desde cualquiera de los dispositivos posibles (terminal o app), se mandará una solicitud al coche disponible más cercana al usuario.

El coche responderá confirmando (o rechazando) la solicitud y su estado cambiará a ocupado.

Una vez el coche termine su servicio, mandará un mensaje al fog avisando de que ha terminado su servicio, i dependiendo de su estado de carga, avisara al fog de que necesita cargar o vuelve a estar disponible.

Aunque los coches se ocupan de comunicar su estado al fog en todo momento, para corregir posibles errores que pudieran ocurrir, el fog preguntara periódicamente a todos los coches para tal de actualizar su estado.

Fog <-> Sensores

Primeramente, el fog mantendrá una lista con todos los sensores del aeropuerto.

Cuando el fog reciba una solicitud que requiera localizar ya sea a un pasajero, grupo de pasajeros, o coche, este enviará una petición a todos los sensores para que busquen al objetivo en cuestión. A partir de todos los datos obtenidos, el fog localizara al objetivo.

Fog <-> Terminal

Cuando un usuario solicite un coche desde una terminal para pasajeros, está enviará una solicitud al fog incluyendo su NIT (número identificador terminal), como la posición de las terminales es fija, el fog mantendrá una tabla terminal - localización i inmediatamente solicitará a un coche de los disponibles que se dirija a la ubicación.

A partir de este momento el funcionamiento es igual que el descrito con los coches.

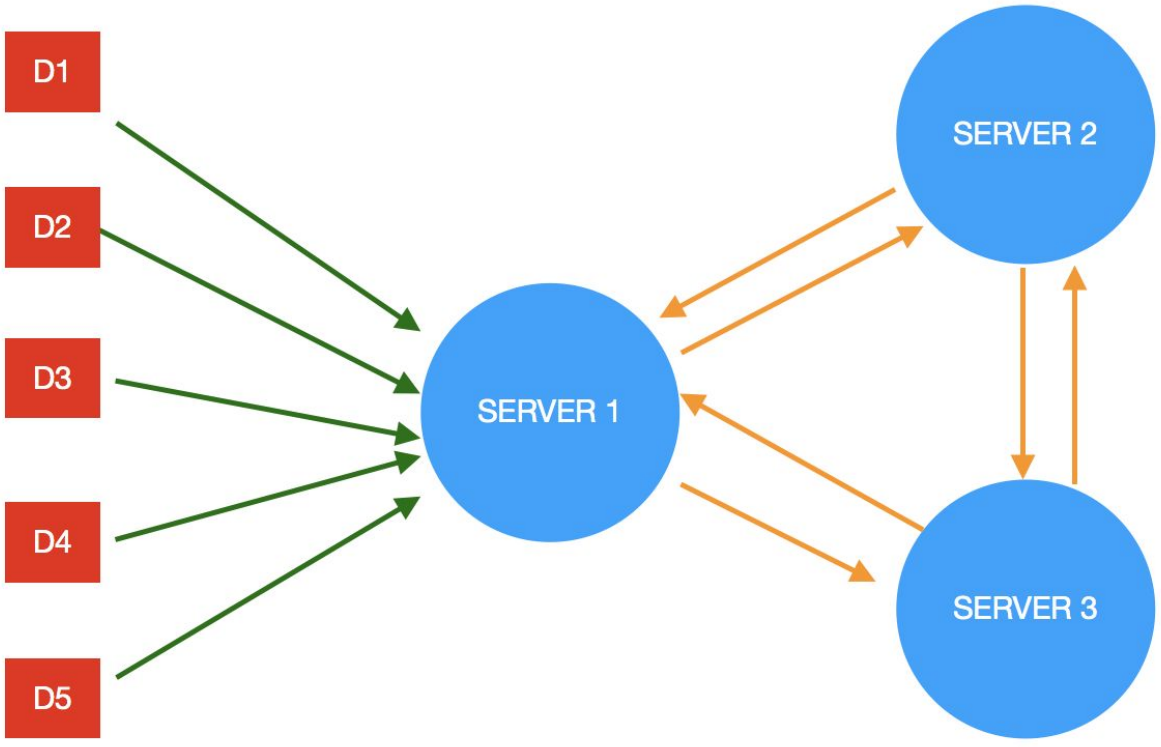
Fog <-> Mobil

Se comunican a través de la app, lo referente a la app se encuentra en su propio apartado.

Primera implementación Fog

Explicacion

De cara a comenzar a implementar algunas partes del sistema, y para obtener algo de feedback sobre la implementación (almenos saber si estamos muy equivocados), hemos realizado una pequeña implementación del FOG, la cual consiste en varios nodos recibiendo y enviando peticiones en forma de sockets y mensajes.



Desde el punto de vista del propio servidor FOG, este puede recibir peticiones desde dos lugares diferentes:

-Peticiones de los dispositivos (edge), la cual puede desencadenar que el servidor realice peticiones a los otros servidores.
-Peticiones de los otros servidores.

Por tal de simplificar temporalmente los requisitos, la primera implementación consistirá únicamente en la comunicación entre los propios servidores.

El objetivo es que cada uno de los tres servidores pueda mandar peticiones a los otros (las cuales consistirán únicamente en responder con un mensaje de texto). En el momento de indicar la petición se abre un socket para gestionar todo lo relacionado a esta, i al finalizar-la se cierra.

A continuación mostramos un esquema de como queda la estructura de un nodo perteneciente al fog en este momento:

En total, en todo momento tenemos un thread esperando conexiones entrantes, otro thread enviando conexiones más un thread para resolver cada petición que recibimos.

Guia Usuario

El archivo ejecutable se encuentra adjunto con el nombre de **nodeFog.py**

En esta primera implementación, los nodos fog tienen que estar en la misma máquina, para comprobar el funcionamiento del codigo se debe hacer lo siguiente:

- Tener python instalado, en principio no importa el SO en el que se ejecute
- Modificar la siguiente linea con la ip de la máquina en la que corre

```
### MODIFICAR LA IP A LA DE LA MAQUINA DONDE CORREREMOS LOS NODOS
ip = '192.168.1.37'
```

Una vez hecho esto, ejecutamos tres nodos en tres terminales diferentes y indicamos los puertos en los que funcionara cada uno, además los puertos en los que funcionarán los otros nodos.

Ahora comparamos el funcionamiento a la hora de enviar peticiones (en forma de mensajes).

De cara a la retroacción nos gustaría saber si la implementación con sockets es correcta o deberíamos utilizar algún otro tipo de tecnología (hemos utilizado sockets porque es lo único que hemos tocado más o menos).

L'aplicació mòbil

Descripció general

MyPassport es una herramienta en forma de aplicación para teléfonos móviles que permite organizar y gestionar el paso por la terminal 2 del aeropuerto. Creando una cuenta en MyPassport podrás consultar los detalles de tu vuelo, acceder a tu tarjeta de embarque, consultar todas las opciones comerciales de la terminal, y entre otras cosas, pedir que uno de nuestros coches autónomos te lleve hasta tu puerta de embarque. Todo esto desde tu propio teléfono móvil.

MODELO DE NEGOCIO

Como se acordó en los términos iniciales, el objetivo principal de esta herramienta es conseguir que el cliente que asiste a la terminal encuentre toda la información accesible en todo momento. Buscamos que el cliente sienta la tranquilidad de saber que tiene todo bajo control. Esta tranquilidad la vamos a traducir en un aumento de la actividad comercial, y por ello la oferta comercial tiene una presencia importante en la app. Le ofrecemos al cliente toda una variedad de actividades comerciales que puede llevar a cabo en la terminal mientras espera su vuelo, sin miedo a llegar tarde a éste.

Explicació software/hardware utilitzat

Nuestro equipo técnico ha decidido que, ya que existe el potencial y la posibilidad entre los miembros del equipo de desarrollar una app en un framework multiplataforma, ésta va a ser desarrollada en React Native.

React Native es el framework de desarrollo de apps multiplataforma más popular y utilizado a día de hoy por las startups, y cobra mucho sentido cuando se lleva a cabo un proyecto con un equipo de personas limitado. Desarrollar una app en este framework te permite tener una app para iOS y para Android sin tener que dedicar dos equipos paralelos a ello. Es suficiente con un proyecto y un equipo dedicado.

Lo mejor de todo es que la experiencia de usuario no se ve afectada, ya que la app luce igual que una aplicación nativa del sistema operativo, a diferencia de lo que pasa con otros frameworks. Será imprescindible la coordinación con los otros equipos a la hora de desarrollar la app, ya que necesitamos, entre muchas otras cosas, saber datos del mapa, bases de datos, API… Por ello los Product Owner tendrán un papel fundamental en el desarrollo de la app.

Manual d'usuari

La app se centra en 5 apartados principales: mapa, comercio, tarjeta de embarque, avisos y perfil.

- **Mapa**: Se trata de un mapa a escala con la localización aproximada a tiempo real del usuario, así como la localización de puntos importantes como establecimientos comerciales, puertas de embarque y coches autónomos
- **Comercio**: La parte de la app que pretende mostrar al usuario todo el abanico de opciones comerciales de las que dispone en la terminal. Desde restaurantes, a tiendas, actividades o zonas reservadas.
- **Tarjeta de embarque**: El cliente tiene acceso a su tarjeta de embarque en todo momento desde este apartado de la app. Podrá usarla para entrar a la terminal mediante un código QR.
- **Avisos**: Canal directo de comunicación entre las aerolineas y los usuarios. El usuario de la app dispondrá de avisos relacionados con su vuelo, y alertas sobre las horas y actualizaciones sobre el vuelo. Las aerolíneas podrán mandar avisos a usuarios en específico.
- **Perfil**: Aquí es desde donde el cliente gestiona sus datos de usuario.

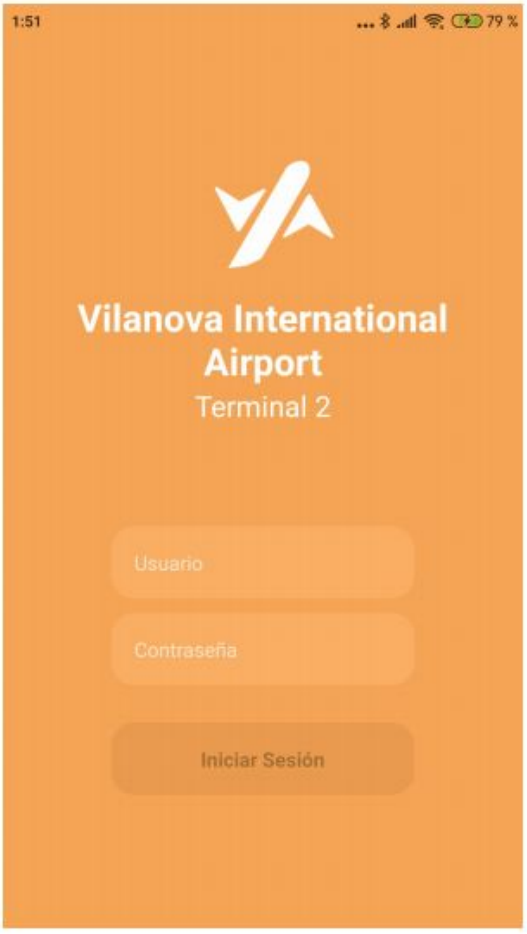
PROGRESO DE LA APP

Inicio de sesión

Para usar la app será imprescindible iniciar sesión con tu usuario.

El usuario es el DNI o número de pasaporte de la persona.

Una vez iniciado sesión podrás acceder a todas las funcionalidades de la app.



Barra de navegación inferior

Se ha diseñado una barra de navegación fácil y intuitiva, que facilita el acceso de los principales apartados de la app.

Mapa

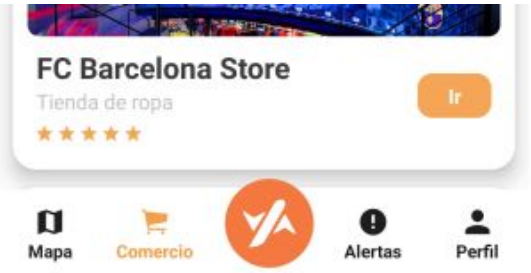
Se han realizado las bases del desarrollo del mapa de la terminal. Éste consistirá en aprovechar los recursos de la API de los mapas de Google, para dibujar sobre éste los modelos del mapa de la terminal.

Se dibujarán también los caminos mediante los recursos que facilita la API.

La localización, se llevará a cabo de forma más compleja, ya que en el interior de la terminal no podemos recurrir a la localización GPS del teléfono. Llevaremos a cabo la localización del usuario a través de la triangulación de la posición del teléfono sabiendo los puntos de acceso de conexión de la terminal que el dispositivo atraviesa mientras se mueve.

Por lo tanto, será imprescindible estar conectado al Wi-Fi de la terminal para consultar la localización o calcular rutas o caminos en la terminal.

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Politècnica Superior d'Enginyeria
de Vilanova i la Geltrú



Tarjeta de embarque

La tarjeta de embarque muestra todos los datos de tu vuelo y incluye un código QR con el que podrás identificarte en el control de la entrada.

Alertas

Las alertas mostrarán información a tiempo real al usuario acerca de novedades en su vuelo, o eventos importantes como por ejemplo aquellos relacionados con los coches autónomos de la terminal.

Perfil

En el perfil encontraremos el pase de identificación personal del usuario que le permitirán acceder a ciertas zonas de la terminal como por ejemplo, zonas VIPS o reservadas o zonas restringidas para trabajadores.

También podrás acceder a un histórico de vuelos, información de la app y gestionar algunas preferencias de la app.

Base de Dades

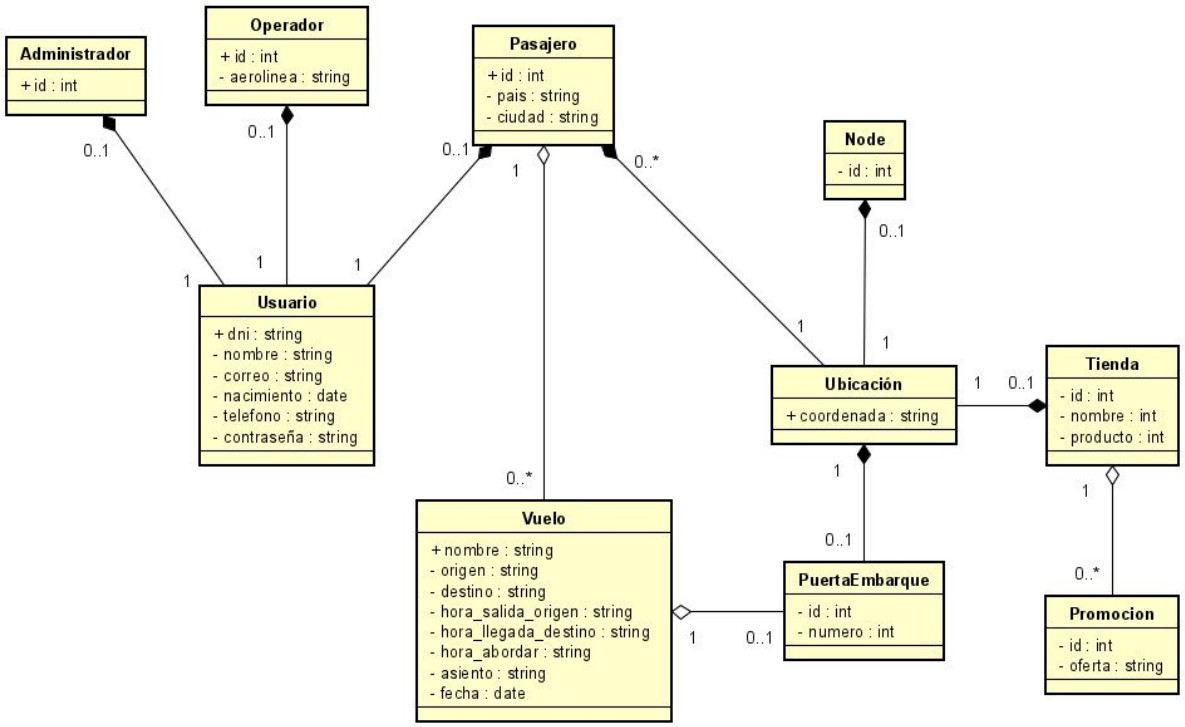
Descripció general

Hemos decidido por optar a un diseño de la base de datos básico. En el esquema que adjuntamos más abajo detallamos como tenemos planeado almacenar la información imprescindible y básica de nuestra Terminal 2 para los diferentes usuarios.

DEFINICIÓN DE LOS CONCEPTOS

- **Usuario**: Persona común que utiliza un sistema informático.
- **Administrador**: Usuario que tiene control absoluto de nuestro sistema inteligente.
- **Operador**: Usuario encargado de gestionar los vuelos de las diferentes aerolíneas.
- **Pasajero**: Usuario cliente al que facilitamos información de sus vuelos y datos.
- **Vuelo**: Trayecto que realiza un avión de un sitio a otro.
- **Puerta_embarque**: Información del lugar donde se accede al avión.
- **Ubicación**: Lugar donde se encuentra alguna cosa(pasajero, puerta de embarque, tiendas, servicio)
- **Node**: Concepto de coche
- **Tienda**: Establecimiento donde se vende algún producto.
- **Promoción**: Publicidad que se da de un determinado producto.

MODELO CONCEPTUAL DEL ESQUEMA



DESCRIPCIÓN MODELO CONCEPTUAL - NO RELACIONAL

Al ser un modelo conceptual no relacional, las líneas que hay entre dos conceptos no indican la relación que hay entre ellos sino como se se desglosa, las flechas con la punta negra indican obligación y las blancas indican que se puede desglosar o no.

Por ejemplo, el concepto pasajero se desglosa en tres conceptos o entidades, pasajero se compone de sus propios atributos, debe tener un usuario, debe tener una ubicación y puede tener un vuelo asignado.

Explicació software/hardware utilitzat

El software que utilizaremos para el almacenamiento de datos es MongoDB, mongo viene de la palabra inglesa humongus(“enorme”), es un sistema de base de datos NOSQL(No relacional).

Lo hemos escogido porque es escalable, flexible, fácil de aprender, usar y sobretodo que la información que almacenaremos será la de un aeropuerto internacional, lo que nos lleva a almacenar grandes cantidades de datos que además irá creciendo constantemente.

DESCRIPCIÓN MODELO LÓGICO NO RELACIONAL EN MONGODB

En las bases de datos no relaciones, el concepto de tabla y su relación mediante claves foráneas son muy diferentes, en mongoDB no existe una estructura de relaciones, en lugar de tablas se utilizan las colecciones(collections) que están formadas con atributos igual que las tablas de las bases de datos relacionales, pero lo que hace que sean no relaciones es que están orientadas a objetos, es decir que dentro de una colección se pueden añadir objetos, es como si en una base de datos SQL se pudiesen añadir tablas dentro de otras tablas, en este apartado se explicará la estructura de cada colección de nuestra base de datos.

Colecciones

passengers

Esta sería una colección de pasajeros, con un id(su documento de identidad). un nombre, el email, su fecha de nacimiento, el país y la ciudad por si en un futuro necesitáramos generar

estadísticas de los pasajeros. También podemos observar que la contraseña de nuestro pasajero está cifrada con un hash y así mantener la privacidad de su contraseña, la ubicación es la posición actual donde se encuentra.
También tiene información respecto a su vuelo y este a su vez tiene información de su puerta de embarque.

Operators

En el caso de los operadores tendría los mismo atributos del concepto Usuario de nuestro MC, solo que en este le añadimos el atributo aerolínea que es para la compañía que trabaja.

Administrators

La colección de Administrators tiene los mismo atributos que el concepto usuario porque hemos supuesto que no habrá muchos administradores por lo tanto no hace falta que tengamos demasiada información de ellos en nuestro sistema para estadísticas o similares.

Shops

Estructura básica de las tiendas, contiene información del nombre, el producto que vende, su localización para guiar a los pasajeros hacia las tiendas cercanas, un atributo llamado type que nos indicará de qué será este comercio, que pueden ser tiendas, restaurantes, de ocio y por último una lista de las ofertas que promocionan.

Node

La colección Node tiene solo estos atributos porque aún no sabemos qué datos relevantes necesitamos.

Flights

Estructura básica de la colección vuelo que guarda información del nombre del vuelo, el origen, destino del vuelo, la hora de embarque, hora a la que llegará a su destino como la hora a la que sale del origen, también tenemos la fecha del vuelo y por último tenemos la información de la puerta de embarque con su nombre y la localización.

Manual d'usuari

Actualment, l'usuari encara no podrà realitzar cap consulta ni revisar el funcionament. El que hem fet en aquest Sprint ha estat començar el disseny i implementació final amb MongoDB. Durant el pròxim Sprint s'acabarà la base de dades al complet i aleshores l'usuari podrà tenir accés a provar les consultes, inserts, etc.