Práctica de computación en la nube

Álvaro Jesús Tapia Muñoz, Jose Luis Bueno Pachón, Carlos Marín Corbera, Carmen González Ortega, Altair Bueno Calvente

30 dic 2022

Contents

Credenciales

```
base de datos: mongodb+srv://root:root@windbnb.uecdw2z.mongodb.net
paypal:
  - email: sb-5qp7q22405408@personal.example.com
    password: 2o}yF$ct
```

Despliegue

Para el despliegue de la parte frontend de la aplicación se ha hecho uso del proveedor Vercel

Para el despliegue de la parte backend de la aplicación se ha hecho uso del proveedor Fly.io

La base de datos se encuentra alojada en un cluster de MongoDB Atlas

- La aplicación se encuentra desplegada en https://windbnb-fawn.vercel.app
- Los microservicios A2viviendasREST, A2datosabiertosREST y A2reservasREST están desplegados respectivamente en:
  - https://a2viviendas.fly.dev
  - https://a2datosabiertos.fly.dev
  - https://a2reservas.fly.dev

Requisitos considerados

Se han considerado los siguientes requisitos para la realización del cliente REST:

- **El almacenamiento de datos se realizará en una base de datos no relacional**
  - Se hace uso de MongoDB Atlas para el almacenamiento de datos
- **Identificación de los usuarios de la aplicación haciendo uso de técnicas basadas en OAuth**
  - Se hace uso de Auth0 para la autenticación y autorización de los usuarios
- **La aplicación permitirá la interacción entre sus usuarios mediante un sistema de comentarios y valoraciones**
  - Se ha actualizado el modelo de la base de datos, añadiendo un nuevo esquema `valoraciones`, que describe una valoración realizada sobre una publicación por un usuario y un comentario
- **Integración de un servicio de pago en la aplicación**:

  - El microservicio `A2ReservasREST` ha sido actualizado con soporte para procesar los pagos de las reservas a través de Paypal.
  - Los distintos métodos de pago incluyen Paypal, Sofort y tarjeta bancaria

- **La aplicación estará desplegada en la nube**

  - Para el despliegue de la parte frontend de la aplicación se ha hecho uso del proveedor Vercel
  - Para el despliegue de la parte backend de la aplicación se ha hecho uso del proveedor Fly.io

Tecnologías Utilizadas

Durante el desarrollo de la interfaz del cliente se han usado los frameworks Astro, Svelte y Bootstrap para la generación de las páginas y componentes usando Typescript. Para los mapas se ha usado la librería Leaflet que hace a su vez uso de OpenStreetMaps. El proveedor OAuth2 escogido es Auth0.

Para la persistencia de datos se ha optado por la base de datos de MongoDB, conectada a través del driver asíncrono Motor, al igual que con el servidor REST, y para el servicio cloud de almacenamiento de imágenes Cloudinary.

Arquitectura de la aplicación y esquema de navegación

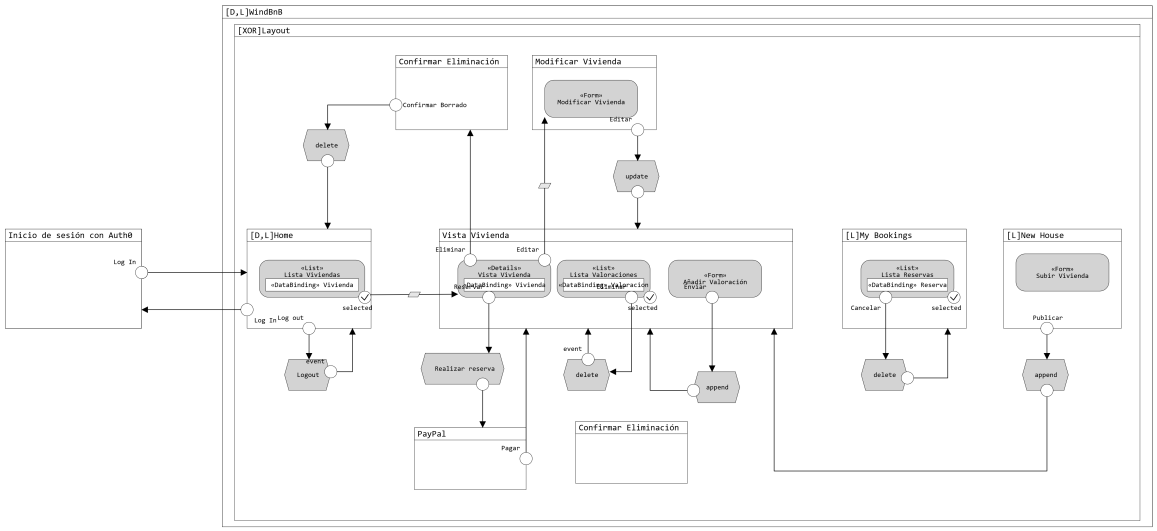

Figure 1: IFML_Model

El esquema de navegación de la aplicación consta de una página principal a la que se puede acceder sin credenciales. Desde ella, el usuario puede iniciar sesión, cambiando a un proveedor externo Auth0, o crear una nueva vivienda (si ha iniciado sesión), o ver sus reservas (si ha iniciado sesión) o ver los detalles de una vivienda en concreto. Dentro de la vista de detalles de la vivienda, el usuario puede eliminar o editar dicha vivienda, siempre que el usuario que esté con la sesión iniciada coincida con el usuario que publicó la vivienda.

Además se puede reservar la vivienda, si el usuario no es el dueño, cambiando a un proveedor externo PayPal para realizar el pago. Dentro de la vista de reservas de un usuario aparece una lista con las reservas, pudiéndose filtrar y cancelar. Dentro de las vistas de creación y edición de una vivienda, aparece un formulario con los campos de dicha vivienda y un botón para realizar la acción correspondiente. Al eliminar una vivienda, se pide confirmación explícita en otra vista.

El modelo de la arquitectura se encuentra en el directorio `/ifml`, junto con unos conjuntos de datos de prueba.

Base de datos

Los documentos almacenados en Mongo mantienen el siguiente esquema:

Colección houses:

```
{
  // Clave primaria de Mongo
  "_id": "ObjectId",
  // Titulo de la vivienda
  "title": "string",
  // Descripción de la vivienda
  "description": "string",
  // Identificador del usuario dueño de la vivienda
  "user_id": "string",
  // Ubicación de la vivienda
  "location": "string",
  // Estado de la vivienda. Por defecto: available
  "state": "enum(available,deleted)",
  //Lista de fotos de la vivienda
  "url_photo": ["string"],
  //Coordenada geográfica longitud
  "longitude": "string",
  //Coordenada geográfica latitud
  "latitude": "string",
  // Lista de todas las reservas que ha recibido esta vivienda
  "bookings": [
    {
      "_id": "ObjectId",
      // Identificador del usuario que hace la reserva
      "user_id": "string",
      // Fecha de inicio de la reserva en formato YYYY-MM-DD
      "start_date": "string",
      // Fecha de fin de la reserva en formato YYYY-MM-DD
      "end_date": "string",
      // Estado de la reserva. Por defecto: reserved
      "state": "enum(reserved,canceled)",
      // Información sobre el pedido de Paypal
      "paypal_order": {}
    }
  ]
}
```

Colección valoraciones:

```
{
  "_id": "ObjectId",
  // Identificador de la vivienda asociada a la valoración
  "vivienda_id": "ObjectId",
  // Identificador del usuario que hace la valoración
  "user_id": "string",
  // Valoración de la vivienda entre 0 y 10
  "valoracion": "integer",
  // Comentario de la vivienda
  "comentario": "string",
  // Estado de la reserva. Por defecto: available
  "state": "enum(available,canceled)"
}
```

Instrucciones de Despliegue

Los todos los microservicios que conforman el backend de la aplicación pueden ser desplegados a través de Docker Compose, o bien desplegarlos individualmente mediante Docker o uvicorn (más información sobre el despliegue de cada uno de los servicios se puede encontrar en el fichero `README.md` en la carpeta de cada microservicio).

El cliente web está configurado para ser desplegado en Vercel, por lo que no se proporciona un método para lanzar el servicio en producción localmente. Para más información sobre como desplegar el servicio en modo desarrollo, visite el fichero `A2clienteREST/README.md`.

Funcionalidad de la aplicación cliente

Página principal `/houses`

- Lista de viviendas que están en la base de datos. Pulsando sobre un título se accede a la página de la vivienda. Muestra varios datos de cada vivienda, almacenados en la base de datos.
- Visualización de una barra de navegación con las opciones de:
  - Home y `WindBnB`: Redirige a la Página principal
  - `Bookings`: Accede a la Páginas de reservas del usuario
  - `New House`: Una vez iniciada sesión, aparece esta opción que redirige a la página para crear una nueva vivienda.
  - `Login`: Redirige a una página para el inicio de sesión.
  - `Logout`: Cierra la sesión del usuario y lo redirige a la página principal.

- Buscador de viviendas mediante un filtro que busca por título de la vivienda.
- Filtro de precio: Se puede filtrar por un rango de precios. (mínimo y máximo).
- Accesible para cualquier usuario, esté autenticado o no.

Página de una vivienda `/houses/{vivienda_id}`

La página de una vivienda tiene distintas funciones:

- Botón `Edit house`: Redirige a una página para completar la acción de editar. Sólo disponible si el usuario es el dueño.
- Botón `Delete house`: Realiza la acción de borrar la vivienda de la base de datos y redirige a la página principal. Sólo disponible si el usuario es el dueño.
- Visualización de **datos** de la vivienda.
- Visualización de **imágenes** de la vivienda.
- Formulario para reservar la vivienda: Muestra dos campos de fecha de calendario para indicar el rango de días que quiere reservar el usuario y varios botones que representan distintos métodos de pago. Al pulsar uno de ellos habiendo seleccionado una fecha de inicio y fin, se iniciará el proceso de pago.
- Visualización de un **mapa**: Muestra la localización de la vivienda a partir de la latitud y longitud almacenados en la base de datos, y las gasolineras cercanas en un área de 5km llamando a la api encargada de los datos abiertos en el backend (`/gas-stations?area={area}&limit={limit}&latitude={latitude}&longitude={longitude}`). Se hace uso de la librería Leaflet y se muestra con OpenStreetMaps.
- Visualización de la **estancia media**: Muestra la estancia media de los viajeros en esa provincia a partir de la provincia de la vivienda llamando a la api encargada de los datos abiertos en el backend (`/average-stay?provincia={provincia}`).
- Accesible para cualquier usuario, esté autenticado o no
  - Los usuarios no autenticados solo podrán visualizar la publicación
  - Los usuarios autenticados que no hayan creado la publicación podrán realizar valoraciones sobre la casa y hacer una reserva
  - El usuario autenticado y autor de la publicación podrá editar y eliminar la misma, pero no podrá realizar valoraciones ni reservas
  - Los usuarios autenticados podrán borrar su propia valoración

Página de reservas `/bookings`

La página de reservas muestra una lista de reservas realizadas por el usuario que está con la sesión iniciada en ese momento. De cada reserva se muestra la vivienda, a la que se puede acceder a su página

pulsando sobre ella, la fecha de inicio y fin de la reserva, el estado y el botón `Cancel booking` cancela la reserva, eliminándola de la base de datos.

Las reservas se encuentran paginadas de 10 en 10, contando con los botones `Previous` y `Next` para navegar entre la lista de reservas del usuario.

También cuenta con opciones de filtro y ordenación de las reservas:

- Filtro por fecha de inicio
- Filtro por fecha de fin
- Filtro por estado
- Ordenar por fecha de inicio o fin
- Orden ascendente o descendente
- Solo accesible para usuarios autenticados

Página de creación de una vivienda `/houses/new`

La página muestra un formulario con los datos correspondientes a una vivienda para crear una nueva vivienda y almacenarla en la base de datos:

- Título
- Descripción
- Calle
- Número
- Ciudad
- Provincia
- Código postal
- País
- Precio por noche
- Imágenes: Mediante un botón `Elegir archivos` se accede al explorador de archivos donde se puede realizar una selección múltiple de imágenes. Es obligatorio seleccionar al menos 1 imagen.

Mediante el botón `Create` se añade a la base de datos con los datos rellenados en el formulario y se redirige a la página de esa vivienda. Las fotos seleccionadas se almacenan en Cloudinary.

- Solo accesible para usuarios autenticados

Página de modificación de una vivienda `/houses/edit/{vivienda_id}`

La página muestra un formulario con los datos correspondientes a una vivienda para modificar una vivienda ya existente en la base de datos, con los campos ya autocompletados:

- Título
- Descripción
- Calle
- Número
- Ciudad
- Provincia
- Código postal
- País
- Precio por noche
- Imágenes

Mediante el botón `Edit data` se actualiza la vivienda de la base de datos si se ha cambiado algún campo y redirige a la página de la vivienda. Si se han modificado imágenes, de la misma manera se actualiza en Cloudinary.

- Solo accesible para un usuario autenticado y autor de la publicación

Conjunto de datos abiertos

El microservicio encargado de servir los datos abiertos es `A2datosabiertosREST`

Precio de carburantes en las gasolineras españolas

El conjunto de datos abiertos relacionado con las gasolineras de España incluye información sobre su posición geográfica, dirección y precio de los carburantes ofertados de cada gasolinera. Este será utilizado para mostrar en el mapa las gasolineras cercanas a la vivienda publicada en una determinada provincia.

Endpoints

- `GET /gas-stations`: Devuelve una lista de gasolineras (`List[EESSPrecio]`) filtrados por provincia y rótulo. Por defecto 10 gasolineras como máximo
  - `provincia`: `Optional[str]`. Nombre de la provincia

  - `rotulo`: `Optional[str]`. Nombre de la marca o rótulo de la gasolinera
  - `limit`: `int`. Número máximo de elementos a devolver. Por defecto, el valor es 10

- `GET /gas-stations/{latitude}/{longitude}`: Devuelve una lista de gasolineras (`List[EESSPrecio]`) que se encuentran como máximo en un área a partir de una geolocalización. Por defecto 10 gasolineras como máximo.

  - `latitude`: `float`. Latitud de la vivienda
  - `longitude`: `float`. Longitud de la vivienda
  - `area`: `int`. Límite del área en kilómetros. Por defecto, el valor es 5
  - `limit`: `int`. Número máximo de elementos a devolver. Por defecto, el valor es 10

El modelo del tipo de salida `EESSPrecio` se puede ver en `A2datosabiertosREST/src/models/gas_stati`
en modo local.

Estancia media de los viajeros por provincias y meses

El conjunto de datos abiertos relacionado con la estancia media de viajeros contiene información de la estancia media en días de los viajeros por provincia y mes según la vivienda.

Endpoints

- `GET /average-stay`: Devuelve el valor de la estancia media (`Data`) de viajeros de una provincia en un mes o año.

  - `provincia`: `str`. Nombre de la provincia
  - `mes`: `Optional[str]`. Nombre del mes

El modelo del tipo de salida `Data` se puede ver en `A2datosabiertosREST/src/models/average_stay.py`
en modo local.

Casos Alternativos

En el caso de que no se introduzca una entrada correcta, se han definido dos excepciones que se elevarán cuando sea oportuno: `NoGasStations` y `NoDataFound`. Si no se encuentran resultados en la búsqueda, el endpoint devuelve un mensaje indicándolo.

NoGasStations Exception

En el caso de `NoGasStations`, se puede mostrar en dos ocasiones. Cuando no hay gasolineras dado una provincia y rótulo se mostraría un objeto

```
{
  "message": "No {rotulo} gas stations found in {provincia}"
}
```

O, en el caso de que no haya gasolineras dado un radio de búsqueda, una latitud y una longitud, se mostraría otro objeto de la misma forma

```
{
  "message": "No gas stations found within {kilometers}km from [{latitude}°,
   ↪  {longitude}°]"
}
```

NoDataFound Exception

Cuando buscamos la estancia media dada una provincia y un mes (o la media anual), puede darse el caso de que no haya datos sobre ello. No es que no se encuentre el recurso, simplemente que no hay información sobre ello. En este caso, se mostrará el objeto

```
{
  "message": "No data was found for [{provincia}, {mes/total}]"
}
```

API REST desarrollada

A2ReservasREST

La siguiente lista es una especificación informal sobre los endpoints REST disponibles en el microservicio, a modo de resumen. La documentación completa se puede encontrar en el propio servidor, bajo las rutas `/docs` (SwaggerUI) y `/redoc` (Redoc). Además, se adjunta una copia local en el fichero `openapi.json`, dentro de la carpeta del proyecto.

> Nota: Todos los endpoints (menos la ruta `ping`) requieren de un token de acceso válido suministrado en la cabecera HTTP `Authorization`. Además, las rutas solo operan **sobre el usuario actual**, por lo que es imposible acceder a las reservas de otros usuarios

- `GET /booking`: Devuelve una lista de reservas que cumplan con los filtros especificados. 10 reservas como máximo. Los parámetros de consulta son:
  - `skip`: Número de reservas a ignorar
  - `house_id`: Identificador de la vivienda
  - `owner_id`: Identificador de usuario (propietario de la vivienda)
  - `before_date`: Fechas de fin anteriores
  - `after_date`: Fechas de inicio posteriores
  - `sort_by`: Campo utilizado para ordenar
  - `ascending`: Ordenar de forma ascendente. Por defecto: `false`
  - `state`: Estado de la reserva
- `POST /booking`: Crea una nueva reserva. El cuerpo de la petición es un json con los siguientes campos:
  - `house_id`: Identificador de la vivienda
  - `start_date`: Fecha de inicio de la reserva
  - `end_date`: Fecha de inicio de la reserva
- `PUT /booking/{booking_id}/capture`: Completa la reserva de una vivienda completando la transacción con Paypal con la siguiente información (parámetros de consulta):
  - `order_id`: Identificador de pedido de Paypal
- `GET /booking/{booking_id}`: Devuelve toda la información sobre la reserva con identificador `booking_id`
- `DELETE /booking/{booking_id}`: Cancela la reserva con identificador `booking_id`
- `GET /ping`: Ruta utilizada para validar que el servicio se encuentra disponible

Casos alternativos

En caso de error en una petición bien formada, se devolverá un mensaje de error siguiendo el siguiente formato.

```
{
  "detail": "error"
}
```

Donde el campo `detail` contiene uno de los siguientes códigos de error

- `ALREADY_BOOKED` (409 Conflict): No se puede reservar por un conflicto
- `NOT_FOUND` (404 Not Found): No se ha encontrado la vivienda
- `UPDATE_ERROR` (404 Not Found): No se ha encontrado la reserva

A2ViviendasREST

Este microservicio se encarga de proporcionar los datos sobre las viviendas.

Endpoints REST disponibles

La siguiente lista es una especificación informal sobre los endpoints REST disponibles en el microservicio, a modo de resumen. La documentación completa se puede encontrar en el propio servidor, bajo las rutas `/docs` (SwaggerUI) y `/redoc` (Redoc). Además, se adjunta una copia local en el fichero `openapi.json`, dentro de la carpeta del proyecto.

> Todos los endpoints (menos las rutas `GET /viviendas` y `/GET /{idCasa}/valoraciones`) requieren de un token de acceso válido suministrado en la cabecera HTTP `Authorization`. Además, las rutas solo operan **sobre el usuario actual**,

- `GET /viviendas`: Devuelve una lista de viviendas que cumplan con los filtros especificados
- `POST /viviendas`: Crea una nueva vivienda. El cuerpo de la petición es un json con los siguientes campos:
  - `title`: Título de la vivienda
  - `description`: Descripción de la vivienda
  - `user_id`: Identificador del usuario que realiza la reserva
  - `location`: Lugar donde se encuentra la vivienda
  - `url_photo`: Fotos de la vivienda
  - `longitude`: Coordenada geográfica longitud de la vivienda
  - `latitude`: Coordenada geográfica latitud de la vivienda
  - `price`: Precio por noche de la vivienda
- `GET /viviendas/{idCasa}`: Devuelve toda la información sobre la vivienda con identificador `idCasa`
- `PUT /viviendas/{idCasa}`: Modifica los campos de la vivienda con identificador `idCasa`
- `DELETE /viviendas/{idCasa}`: Borra la vivienda con identificador `idCasa`
- `GET /viviendas/{idCasa}/getBookingsAmount`: Devuelve la cantidad de reservas de una vivienda con identificador `idCasa`
- `GET /{idCasa}/valoraciones`: Devuelve una lista de valoraciones de una vivienda con identificador `idCasa`
- `POST /{idCasa}/valoraciones`: Crea una nueva valoración de una vivienda con identificador `idCasa` con los siguientes campos:
  - `user_id`: Identificador del usuario que realiza la valoración

    - `valoración`: Puntuación de la vivienda(del 1 al 10)
    - `comentario`: Comentario de la vivienda

- `DELETE /valoraciones/{idValoracion}`: Borra la valoración con identificador `idValoracion`

Casos alternativos

En el caso de no encontrar una vivienda con el identificador proporcionado, se devolverá una excepción de tipo NotFoundError con su mensaje correspondiente dependiendo de la operación.

```
{
  "error_code": "string"
}
```