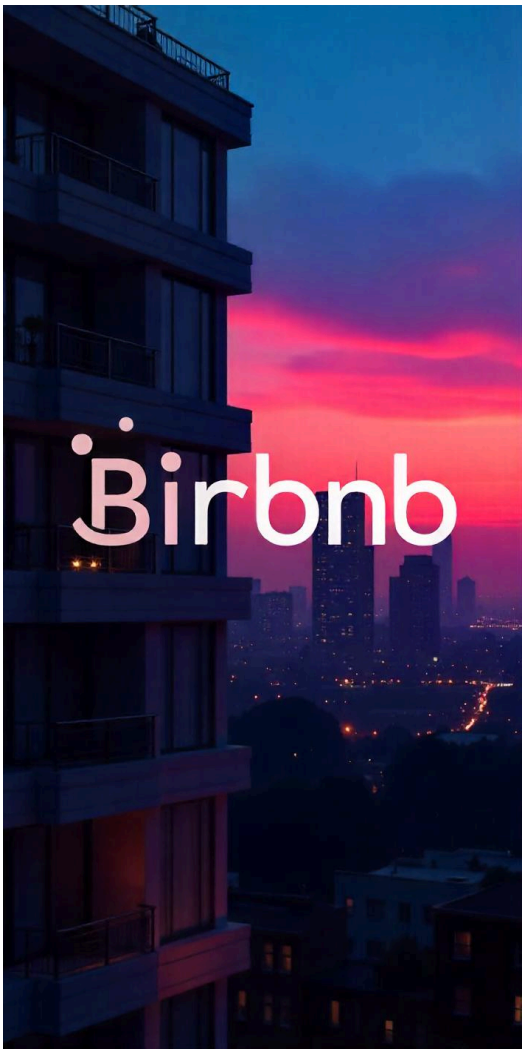
UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES
CÁTEDRA DESARROLLO DE SOFTWARE

Birbnb - Plataforma de Reservas de Alojamientos



Trabajo Práctico Integrador
-1C 2025-

Contexto general

Debido a la creciente demanda de alojamiento no convencional, como departamentos, cabañas y casas particulares en lugar de hoteles tradicionales, ha surgido la necesidad de desarrollar una plataforma moderna e intuitiva que facilite la gestión de reservas de estos espacios. Cada vez más viajeros buscan experiencias únicas y flexibles, optando por alojamientos que les brinden mayor comodidad, privacidad y opciones personalizadas, ya sea para turismo, trabajo remoto o estadías temporales.

En respuesta a esta tendencia, una empresa emergente en el sector del turismo ha identificado la oportunidad de desarrollar un Sistema eficiente que permita a los propietarios registrar y administrar sus inmuebles, mientras que los usuarios interesados puedan realizar búsquedas, visualizar detalles y efectuar reservas de manera sencilla y segura. Además, se busca que la plataforma cuente con un sistema de notificaciones, asegurando que tanto propietarios como huéspedes se mantengan informados sobre el estado de sus reservas y otros eventos importantes.

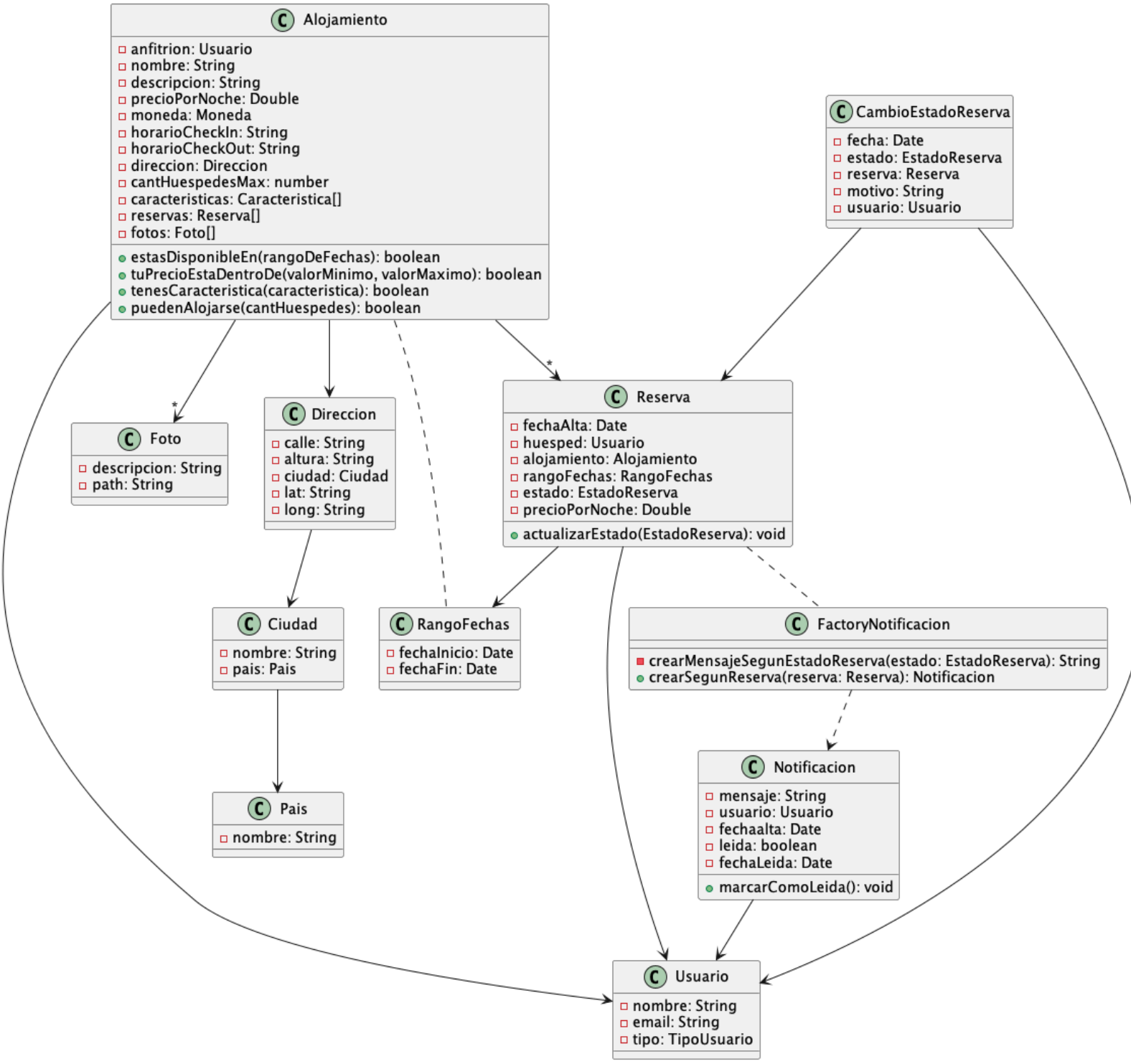
Para lograr este objetivo, diseñaremos e implementaremos un Sistema de Reservas llamado ***Birbnb***, que proporcionará una experiencia fluida y confiable para la gestión de alojamientos temporales.

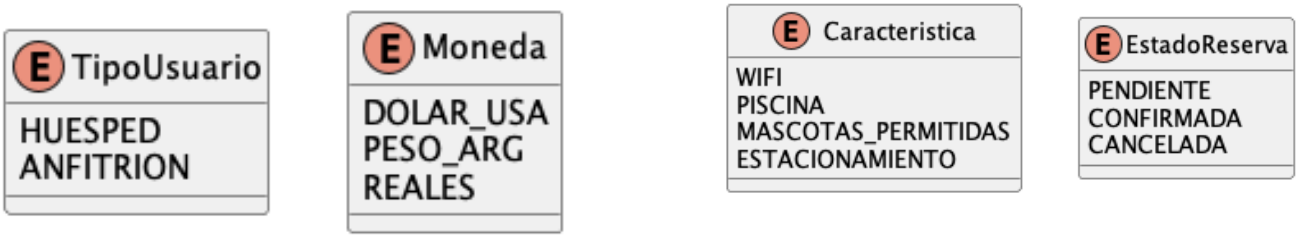
El desarrollo del sistema se llevará a cabo en cuatro fases, cada una enfocada en una parte clave de la implementación.

Entrega 1: Implementación del Modelo de Objetos + Configuración del Proyecto

Contexto

En esta primera iteración nos enfocaremos en implementar el diseño del modelo de objetos provisto por el Teach Leader, que por cierto consideramos correcto. Será nuestra responsabilidad implementar cada uno de los métodos mencionados en el Diagrama de clases.





Respecto a las notificaciones, nuestro Teach Leader nos ha mencionado que:

- Cada vez que se realice una reserva es necesario enviarle una notificación al Anfitrión, donde se le indique quién realizó la reserva, para cuándo, por cuántos días y sobre qué alojamiento.
- Cada vez que un Anfitrión acepte una reserva es necesario enviarle una notificación al huésped, para darle aviso de que su reserva fue confirmada por parte del Anfitrión.
- Si un huésped decide cancelar una reserva, es necesario enviarle una notificación al Anfitrión para darle aviso de este hecho, contándole el motivo de la cancelación si es que fue especificado.

Por otra parte, también debemos implementar un primer endpoint que nos permita verificar el estado de salud de nuestro Sistema (el famoso “/health”).

Entregables

1. ***Implementación*** del Diagrama de Clases provisto, completando cada uno de los métodos mencionados y agregando aquellos que el equipo considere necesarios.
2. ***Implementación*** del endpoint “Health Check”.
3. ***Explicación*** del Git flow definido para utilizar a lo largo del proyecto.

Tecnologías a utilizar

- Backend
  - Lenguaje de programación base: JavaScript
  - Entorno de ejecución: Node.js
  - Framework: Express
- Git como Sistema de Versionado de Código, con repositorios en Github.

Entrega 2: Exposición de APIs + Persistencia

Contexto

En esta segunda iteración debemos desarrollar los siguientes endpoints necesarios para garantizar el cumplimiento de los requerimientos listados a continuación, teniendo en cuenta que todos los datos que se manipulen deben ser persistentes. Además, nuestro Teach Leader nos ha solicitado que la API se base en el enfoque REST.

Gestión de Reservas

La plataforma debe permitir a los usuarios realizar reservas sobre los alojamientos disponibles. Esto implica la implementación de endpoints que gestionen el ciclo de vida de una reserva:

- Creación de una reserva, asegurando la disponibilidad del alojamiento en las fechas seleccionadas.
- Cancelación de una reserva antes de su fecha de inicio.
- Consulta del historial de reservas de un usuario.
- Modificación de una reserva dentro de las reglas establecidas por el Sistema (por ejemplo, cambios de fechas si el alojamiento sigue disponible).

Búsqueda de Alojamientos

Los usuarios deben poder realizar búsquedas de alojamientos en función de diferentes criterios para encontrar el lugar que mejor se adapte a sus necesidades. Se espera:

- Un endpoint para listar alojamientos disponibles, con la posibilidad de aplicar filtros como:
  - Ubicación (ciudad, país, coordenadas, etc.).
  - Rango de precios.
  - Cantidad de huéspedes permitidos.
  - Características especiales (Wi-Fi, piscina, mascotas permitidas, etc.).
- Implementación de paginación para mejorar la eficiencia de las consultas.

Visualización de notificaciones

Los usuarios deben recibir información relevante a través de notificaciones dentro de la plataforma. Las notificaciones estarán relacionadas con:

- Confirmación de reserva.
- Recordatorios de check-in y check-out.
- Cambios en el estado de una reserva (aprobada, rechazada, cancelada).

Se espera:

- Endpoint para obtener la lista de notificaciones sin leer de un usuario.
- Endpoint para obtener la lista de notificaciones leídas de un usuario.
- Endpoint para marcar una notificación como leída.

Por otro lado, también es necesario implementar los tests unitarios sobre la capa de Servicios del Sistema.

Entregables

1. ***Implementación*** de la API REST del Sistema, que incluya todos los endpoints necesarios para dar solución a los requerimientos listados.
2. ***Implementación*** de la persistencia de las entidades de dominio en una base de datos NoSQL Documental.
3. ***Implementación*** de los Test Unitarios de la capa de servicios y de la capa de dominio.
4. ***Documentación*** de la API REST.

Tecnologías a utilizar

- Todas las mencionadas en la primera iteración.
- MongoDB como Base de Datos Documental NoSQL.
- Jest como framework de Testing Unitario.
- Swagger como Herramienta de Documentación de APIs. Está permitido la utilización de alguna dependencia que genere el contenido de Swagger.

Entrega 3: UI + Integración con Backend

Contexto

En esta tercera iteración nos enfocaremos en el desarrollo inicial del Frontend de nuestro aplicativo. En particular, trabajaremos en diseñar, maquetar e implementar las pantallas, con la correspondiente navegabilidad entre ellas, pero solamente integrando una de las funcionalidades (por ahora) con nuestro Backend.

Nuestro Teach Leader nos ha dicho que tenemos la libertad de generar las interfaces de usuario según nuestro gusto y criterio, pero teniendo en cuenta que es importante respetar los siguientes requerimientos no funcionales:

Interfaz Intuitiva

- El diseño de la aplicación debe ser claro y fácil de entender para usuarios sin experiencia previa en plataformas similares.
- Se debe utilizar una estructura de navegación coherente, asegurando que los usuarios puedan acceder a las funcionalidades principales en pocos clics.

Aprendizaje Rápido

- La interfaz debe seguir patrones de diseño comunes en aplicaciones de reservas para que los usuarios reconozcan fácilmente las funcionalidades.
- Se deben incluir textos descriptivos y elementos visuales que guíen a los usuarios en cada paso del proceso de búsqueda.

Feedback Visual y Notificaciones

- Deben incluirse mensajes de error y confirmación en cada interacción relevante, como la aplicación de filtros o la selección de una propiedad.
- Indicadores visuales (loaders, skeletons, etc.) deben mostrar el estado de carga de las solicitudes al Backend para evitar que el usuario piense que la aplicación está inactiva.

Diseño Responsivo

- El aplicativo debe ser completamente funcional en distintos dispositivos (desktop, tablets y móviles).
- Se debe asegurar que todos los elementos sean accesibles en pantallas de distintos tamaños sin afectar la usabilidad.

Accesibilidad

- Se deben seguir buenas prácticas de accesibilidad, como el uso de contrastes adecuados, etiquetas ARIA y soporte para navegación mediante teclado.
- Los textos y botones deben ser lo suficientemente grandes para facilitar su uso en dispositivos móviles.

Consistencia en la UI

- Los estilos, colores y tipografías deben ser homogéneos en toda la aplicación para mejorar la experiencia del usuario.
- Los componentes reutilizables deben seguir un diseño coherente para evitar confusión.

Por otro lado, tal como fue mencionado, tendremos que implementar de forma completa la funcionalidad de "*Búsqueda de Alojamientos*", realizando las correspondientes llamadas a nuestro Backend.

Entregables

1. ***Maquetado*** de todas las pantallas que el equipo considere necesarias para dar cumplimiento a todos los requerimientos funcionales de la iteración 2, con la correspondiente navegabilidad entre ellas y ajuste al cumplimiento de los requerimientos no funcionales listados en esta iteración.
2. ***Implementación*** de funcionalidad completa de "*Búsqueda de Alojamientos*", incluyendo su integración con el backend.
3. ***Justificación*** acerca del cumplimiento de los requerimientos no funcionales.

Tecnologías a utilizar

- Todas las mencionadas en las anteriores entregas.
- Next.js como Framework para el desarrollo de nuestro Frontend.
- React como biblioteca de generación de componentes para nuestro Frontend.
- Axios como Cliente HTTP para integrar nuestro Frontend con nuestro Backend.

Entrega 4: Entrega final

Contexto

En esta última iteración nos enfocaremos en terminar de implementar todas las funcionalidades de nuestro aplicativo, integrando de forma completa el Frontend con el Backend. Además, también tendremos que ocuparnos de realizar Tests de Integración en la capa de controladores de nuestro Backend, y Test E2E en nuestro Frontend.

Por último, nuestro Teach Leader nos ha indicado que llegó el momento de poner en producción la primera versión del aplicativo para que esté disponible en la Web para todos los usuarios, motivo por el cual tendremos que ocuparnos de desplegar la solución en nube.

Entregables

1. ***Implementación*** de todas las funcionalidades completas, con la correspondiente integración con el Backend.
2. ***Despliegue*** del aplicativo en la nube (Backend + Frontend).
3. ***Documentación*** acerca del despliegue del aplicativo que contenga el detalle de los pasos a seguir para desplegar el aplicativo por primera vez y por cada vez que se quiere subir a producción una nueva release.
4. ***Tests de integración*** en la Capa de Controladores del Backend: solamente es necesario realizar 1 test, cuyo endpoint a testear queda a criterio del equipo.
5. ***Tests E2E*** en el Frontend: solamente es necesario realizar 1 test, cuya funcionalidad a testear queda a criterio del equipo.

Tecnologías a utilizar

- Todas las mencionadas en las anteriores entregas.
- Render como Cloud Application Platform para el despliegue de nuestro Backend (se pueden utilizar algunas otras alternativas, a criterio del equipo y en común acuerdo con el equipo docente).
- Netlify como plataforma para despliegue de nuestro Frontend (se pueden utilizar algunas otras alternativas, a criterio del equipo y en común acuerdo con el equipo docente).
- Cypress como herramienta de Testing E2E. Está permitido buscar y utilizar alguna otra herramienta alternativa (como Jest, por ejemplo).
- Jest como framework para Testing de Integración