C/ Societat Unió Musical, 8 - 03802 Alcoi
Tel. 966 52 76 60 - Fax 966 52 76 61
03012165.secret@gva.es
www.cipfpbatoi.es

Proyecto 4 - BatBatCar (con BBDD e interfaces gráficas)

Enunciado

En esta actividad vamos a realizar la **versión con interfaz gráfica** de la aplicación (proyecto) `BatBatCar` desarrollada en las unidades 8 y 9.

A través de la interfaz gráfica se pretende resolver la mayoría de los casos de uso propuestos en el *Proyecto BatBatCar* realizado en unidades anteriores. Sin embargo, para reducir complejidad a la aplicación, **se ha eliminado:**

- **El caso de uso *Login o Establecer Usuario***. Esto implica que en las creaciones de viajes y reservas se preguntará por el nombre de usuario asociado ya que no habrá un usuario en la sesión.
- **La tipología de los viajes.** En este caso, la aplicación pasa a trabajar sólo con viajes del tipo Estándar (ya no habrá ni viajes cancelables, ni flexibles, ni exclusivos), al cual se le podrán aplicar todas las operaciones posibles sin restricción alguna (crear reservas, cancelar viaje, cancelar reservas, modificar reservas...)

Para el desarrollo de esta actividad se indican las pautas necesarias para su buena realización y que deberás respetar. No obstante, se indican diseños de vistas a las que deberás proporcionarle tu diseño propio. Del mismo modo, se permite hacer modificaciones si no afecta a la idea general planteada.

Antes de ponerte a trabajar, debes partir de la siguiente plantilla. **Crea un repositorio propio con una copia de ella para empezar a trabajar en local con tu entorno de desarrollo.**

Se ha distribuido la práctica en dos fases:

C/ Societat Unió Musical, 8 - 03802 Alcoi
Tel. 966 52 76 60 - Fax 966 52 76 61
03012165.secret@gva.es
www.cipfpbatoi.es

FASE 1: Primeras vistas

trabajando sobre datos en memoria

Lo primero que debes hacer es **crear una nueva rama "fase1"** en tu repositorio y empezar a trabajar en ella (no olvides ir haciendo commits conforme vayas logrando ciertos objetivos). En esta primera fase, **nuestra vista principal será el listado de todos los viajes**. A partir de esta vista podremos:

- **Dar de alta** un nuevo viaje
- **Ver el detalle** de un viaje
- **Ver las reservas** de un viaje
- **Dar de alta nuevas reservas** en un viaje.

Debes partir de esta plantilla. Ésta dispone ya de todos los métodos de los DAO que trabajan con los datos almacenados en memoria (`InMemoryViajeDAO` y `InMemoryReservaDAO`) necesarios para que todas estas funcionalidades puedan desarrollarse con éxito. También dispones de algunos de los métodos del repositorio en la clase `ViajesRepository`. También está implementado parte del caso de uso principal (*Listar todos los viajes*). Por tanto, tu trabajo en esta primera fase es:

- Finalización del caso de uso *Listar todos los viajes*.
- Creación de las vistas adicionales y dotar de funcionalidad a las mismas a través de la creación de recursos (endpoints) en el controlador correspondiente.
- Para los dos puntos anteriores, necesitarás implementar los métodos del repositorio que vayas necesitando (antes de crear un método nuevo en el repositorio, comprueba que sea uno de los ya definidos).

Modelo (capa de acceso a datos)

Como se ha comentado anteriormente, en esta primera fase, la aplicación hará uso de una capa de acceso a datos donde el origen de dichos datos serán datos en memoria a través de los DAO `InMemoryViajeDAO` y `InMemoryReservaDAO`. De estos DAO no debes preocuparte ya que están todos los métodos implementados para llevar a cabo esta primera fase. Ahora bien, deberás implementar/crear los métodos de `ViajesRepository` que vayas necesitando a medida

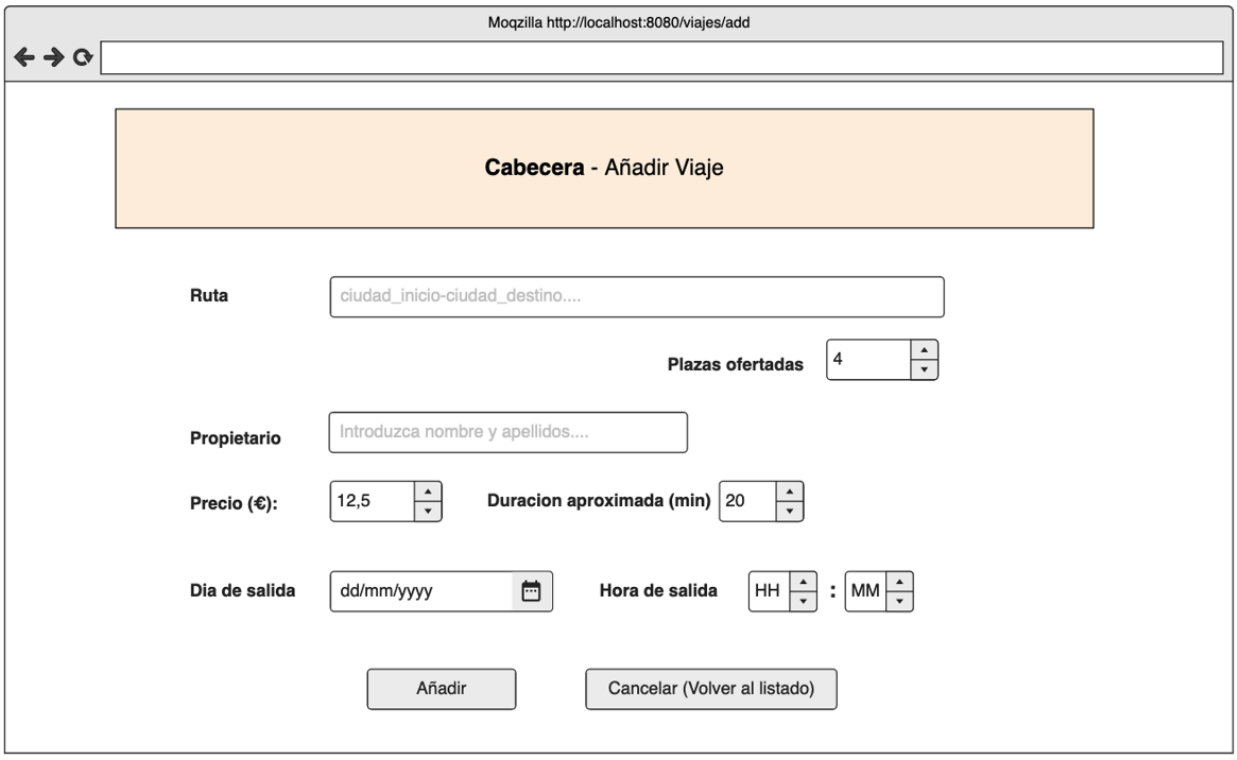
que vayas implementando casos de uso.

Vistas y controladores (casos de uso)

La aplicación deberá disponer de los siguientes controladores con los recursos (*endpoints*) que se detallan a continuación.

`ViajesController`

1. **Añadir un nuevo viaje (`/viaje/add`):** Permitirá añadir un viaje. Deberás tener en cuenta que necesitas implementar tanto el método POST para procesar el formulario como el método GET para mostrar el formulario y que deberás validar los datos (se indica más abajo la validación a llevar a cabo). Se muestra un diseño del formulario:



- El campo "Día de salida" será un input de tipo `date`. Como este componente sólo recoge una fecha sin tiempo, se deberá añadir otro input de tipo time para recoger la hora y minutos (este último puedes hacerlo también con 2 inputs de tipo `number`).
- El campo "Plazas ofertadas" será un input de tipo `number` con un min de 1 y un máximo de 6.
- El botón "Añadir" enviará el formulario al recurso de añadir viaje utilizando el método POST, el cual validará todos los datos introducidos:
  - El campo Ruta debe tener información y además cumplir con que cumpla con formato *Origen – Destino*

- ➢ *Las plazas ofertadas debe ser un valor mayor a 0*
- ➢ *El propietario debe contener al menos dos cadenas separadas por espacio en blanco, donde el inicio de cada una de las cadenas debe comenzar por mayúscula.*
- ➢ *El precio debe ser un valor mayor a 0 y admitir decimales*
- ➢ *La duración debe ser un valor mayor a 0*
- ➢ El día y hora de salida debe estar introducido
- ➢ Si la validación es correcta, añadirá el viaje a la base de datos y mostrará todo el listado de viajes en el que debe aparecer la nueva reserva.
- ➢ Si la validación es incorrecta, se mostrará un mensaje de error en la misma vista del formulario.

❖ El botón "Atrás", será un enlace a la página de mostrar todos los viajes (siguiente punto). Puedes utilizar un elemento de tipo `button`, cambiando su tipo por defecto (el tipo por defecto es `submit`). Puedes utilizar el siguiente snippet de código.

```
<a href="url"><button type="button">Volver al listado</button></a>
```

❖ La gestión de mensajes de confirmación y de mensajes de error debe ser la habitual que se ha trabajado en clase.

2. **Listar viajes (`/viajes`)**: recibirá como **parámetro opcional** el lugar al que se quiere ir. Si este parámetro está presente solo se mostrarán los viajes que van a ese lugar, si no existe dicho parámetro se mostrarán todos sin filtro. Todos los viajes se mostrarán en una tabla similar a la mostrada. Fíjate que, si el viaje que representa está en estado Abierto, deberá mostrar un enlace (Reservar) al **caso de uso 1** (*reservar sobre un determinado viaje*).

3. **Mostrar detalle del viaje (`/viaje`)**: recibirá como **parámetro obligatorio** el viaje del que se quiere consultar la información y mostrará todos sus datos. El formato de visualización se deja a elección del alumno, aunque se aconseja utilizar el formulario con el que se añade un viaje (preferentemente compartiendo esta vista), pero en este caso aparecerán todos los datos del viaje precargados y sin posibilidad de edición (*readonly*). Añade **un enlace** que retorne a la vista de listado de viajes (punto anterior).

`ReservaController`

4. **Añadir reserva de un determinado viaje (`/viaje/reserva/add`):** Permitirá añadir una reserva de un determinado viaje. Deberás tener en cuenta que necesitas implementar tanto el método POST para procesar el formulario como el método GET para mostrar el formulario. El método GET recibirá como parámetro el **código del viaje** sobre el que se añadirá la reserva, en caso de no recibir dicho dato, se redirigirá automáticamente a la página donde se listan todos los viajes. Se muestra un diseño de la vista asociada a este recurso:

Si se puede realizar la reserva, se retornará a la vista de lista de viajes y se mostrará un mensaje de confirmación. En caso contrario, se permanecerá en la vista actual mostrando un mensaje de error

A la hora de crear la reserva, recuerda que las restricciones para la creación de una reserva son:

- que el usuario que hace reserva no sea igual al usuario propietario
- que el viaje esté abierto
- que el viaje no esté cancelado
- que haya plazas suficientes
- que el usuario no haya realizado una reserva anterior ya en ese viaje

Se aconseja la realización de un método en ViajesRepository que se llame:

```
findViajeSiPermiteReserva(codViaje, usuario, plazasSolicitadas)
```

Este método tendrá que recuperar el viaje asociado a la reserva si es que en ese viaje se permite realizar esa reserva. De otro modo lanzará una excepción.

Ya en el controlador, la creación de la reserva se llevará a cabo si el viaje pudo ser recuperado.

5. **Listar reservas de un determinado viaje** (**`/viaje/reservas`**): recibirá como parámetro opcional el código del viaje sobre el que se quieren consultar las reservas y mostrará un

listado de todas sus reservas. En caso de no recibir el código de viaje, se redirigirá al usuario a la página de listar todos los viajes.

FASE 2 Implementación de SQLViajeDAO y SQLReservaDAO para hacer funcionar lo desarrollado en fase 1 y nuevas funcionalidades

Debes **crear una nueva rama "fase2"** a partir de la rama "fase1". A partir de aquí vamos a comenzar con esta nueva fase:

Modelo (capa de acceso a datos)

En esta segunda fase implementaremos dos nuevos DAO (`SQLViajeDAO` y `SQLReservaDAO`) que implementarán las mismas *interfaces* que los DAO utilizados en la fase 1.

La primera acción que tendrás que llevar a cabo en esta segunda fase, es **que tu aplicación funcione exactamente igual que lo hacía al acabar la primera fase**, pero esta vez, el origen de datos será la base de datos `batbatcar` sobre los que actúan los nuevos DAO creados. La base de datos la tienes disponible en la carpeta `resources/database` de la plantilla.

Ten en cuenta que, para esta segunda fase, necesitarás ampliar también tu modelo (entidades, dao's, repositorios...).

C/ Societat Unió Musical, 8 - 03802 Alcoi
Tel. 966 52 76 60 - Fax 966 52 76 61
03012165.secret@gva.es
www.cipfpbatoi.es

Vistas y controladores

La aplicación deberá disponer de los siguientes controladores con las vistas y recursos que se detallan a continuación:

1. **Mostrar detalle del viaje (`/viaje`)**: A la vista de detalle ya desarrollada en la fase 1, se añadirá:
   a. Dos botones que nos permitan:
      i. Cancelar el viaje
      ii. Realizar una reserva, Este botón solo se activará cuando se puedan añadir reservas a un viaje, es decir se encuentra abierto, queden plazas disponibles y no haya salido.
   b. El listado de todas las reservas realizadas de ese viaje (si es que tiene alguna). Cada reserva dispondrá de un botón de cancelar y otro para su detalle.

2. **Listar viajes (`/viajes`)**: A la vista ya realizada en la fase 1, añade:
   a. **2 columnas** que nos permitan saber el **número de reservas** realizadas y el **número de plazas disponibles.**
   b. Ya que no existe la tipología de viaje, vamos a permitir que los viajes se puedan cancelar. **Añade un enlace que nos permita cancelar cada viaje** (si todavía está abierto y no ha tenido lugar)**.**
   c. Un **buscador (formulario) en la parte superior** que nos permita buscar un viaje según un determinado destino.

C/ Societat Unió Musical, 8 - 03802 Alcoi
Tel. 966 52 76 60 - Fax 966 52 76 61
03012165.secret@gva.es
www.cipfpbatoi.es

3. **Ver el detalle de una reserva (`/viaje/reserva`)**. Mostrará todos los datos de la reserva (usuario y número de plazas), además debe aparecer en la vista el viaje al que pertenece. El formato de visualización se deja a elección del alumno, aunque se aconseja utilizar el formulario con el que se añade una reserva (preferentemente compartiendo esta vista), pero en este caso aparecerán todos los datos de la reserva precargados y sin posibilidad de edición (*readonly*). Añade **un enlace** que retorne a la vista de listado de reservas del viaje asociado a esa reserva.

4. **Ampliar listado reservas de un determinado viaje** (**`/viaje/reservas`**): Añade al listado de reservas un enlace en cada reserva para que se acceda al detalle de cada una de ellas.