1. Desarrollo api-rest
   1. Planteamiento inicial.

Como indicaba nuestro plan de proyecto, la semana cuatro comenzamos con el planteamiento de la api, principalmente de los métodos que debíamos implementar y de la estructura de tendría.

Lo primero que hicimos fue estipular la jerarquía de datos para crear las uris de acceso a los mismos, en nuestro caso, había dos conjuntos de datos que están directamente relacionados. Por un lado los coches ofertados dependen de la filial donde se encuentran, por otro, las reservas y los pagos depende del usuario que los realiza. Esto nos deja con las siguientes uris de acceso a los datos:

* Usuarios: <http://localhost:8080/apirest/api/usuario>
* Pagos: [http://localhost:8080/apirest/api/usuario/{idUsuario}/pago](http://localhost:8080/apirest/api/usuario/%7bidUsuario%7d/pago)
* Reserva:  [http://localhost:8080/apirest/api /usuario/{idUsuario}/reserva](http://localhost:8080/usuario/%7bidUsuario%7d/reserva)
* Filiales: http://localhost:8080/apirest/api/filial
* Coches ofertados: [http://localhost:8080/apirest/api/filial/{idFilial}/oferta](http://localhost:8080/apirest/api/filial/%7bidFilial%7d/oferta)

Siendo localhost:8080 el lugar donde se encontrará el servicio en nuestro caso, y /apirest/api el nombre que le daremos posteriormente a la api. En este caso, para la primera versión de la aplicación y para hacer las pruebas, pensamos que era mejor desarrollarlo todo el local y en un futuro hacer pública la api. Una vez conocida la organización de la api y teniendo una orientación de los métodos a desarrollar, gracias a los requisitos que hemos comentado al principio de este archivo, podíamos empezar a desarrollarla.

* 1. Problemas encontrados.

Al ser algo que nunca antes habíamos hecho y que lo único que teníamos en claro era que queríamos hacerlo con Java, buscamos información en internet sobre cómo hacerlo.

Al principio seguimos una guía que usaba un web-logic server de Oracle y dynamic web project de eclipse creado con maven para las dependencias, sin embargo, a pesar de que conseguimos ponerlo en funcionamiento, fuimos incapaces de configurarlo para que procesara y devolviera los objetos en formato json por incompatibilidad, ausencia de alguna versión de las dependencias, buscamos información para intentar arreglar el problema, pero fuimos incapaces. Fue entonces cuando recordamos que en la asignatura de Sistemas Orientados a Servicios usábamos un servidor tomcat, pero al importar el servidor para usarlo en eclipse teníamos que usar una versión de hace cuatro años por algún motivo, porque las nuevas no soportaban tomcat superiores al 8.

* 1. Despliegue del servicio

Al final, tenemos el servidor corriendo en un kernel de Linux dentro de una maquina Windows, de esta manera podíamos compilar los archivos, moverlos y desplegar el servicio de una manera más rápida, además, las variables de entorno de Linux nos facilitaban el hecho de asegurarnos de que el servidor usara el jdk de java que nosotros necesitamos, en este caso el 1.8, ya que, en caso de usar otra versión, el servidor no se ejecutaba o causaba problemas de compatibilidad con las librerías.

* 1. Modelos de datos.
* Filial:

Título: filial

Tipo: objeto

Descripción: Objeto que contiene la localización de una filial

Requerido:

idFilial

href

comunidad

calle

Atributos:

idFilial:

Tipo: int

Descripción: Identificador único para esta filial

href:

Tipo: String

Descripción: Uri de acceso al recurso dentro de la api

comunidad:

Tipo: String

Descripción: Nombre de la comunidad autónoma donde se encuentra esta filial

ciudad:

Tipo: String

Descripción: Ciudad donde se encuentra la filial

calle:

Tipo: String

Descripción: Calle donde se encuentra la filial

numero:

Tipo: int

Descripción: Numero de la calle donde se encuentra, aunque no siempre la habrá

* CocheOfertado:

Título: CocheOfertado

Tipo: objeto

Descripción: Objeto que contiene todos los datos de un coche

Requerido:

idCoche

href

matricula

marca

numeroPuertas

capacidadMaletero

cambioMarchas

plazas

aireAcondicionado

precioDia

kilometraje

politicaCombustible

politicaCancelacion

hrefFilial

modelo

Atributos:

idCoche:

Tipo: int

Descripción: Identificador único para cada coche

href:

Tipo: String

Descripción: Uri de acceso al recurso dentro de la api

matricula:

Tipo: String

Descripción: Dato de siete caracteres que indica la matrícula del coche, única para cada uno

marca:

Tipo: String

Descripción: Marca del coche

numeroPuertas:

Tipo: int

Descripción: Numero de puertas del coche

capacidadMaletero:

Tipo: int

Descripción: Espacio del maletero medido en maletas medianas

cambioMarchas:

Tipo: String

Descripción: Esta variable tomara dos valores, automático o manual

plazas:

Tipo: int

Descripción: Cantidad de ocupantes que puede tener el coche

aireAcondicionado:

Tipo: boolean

Descripción: Esta variable será True en caso de tener aire acondicionado y False en caso contrario.

precioDia:

Tipo: double

Descripción: Esta variable indicara el precio por dia reservado del coche

kilometraje:

Tipo: int

Descripción: Cantidad de kilómetros que puede recorrer el cliente con el coche dentro de la oferta, en caso de pasarse se le cobrara mas

politicaCombustible:

Tipo: String

Descripción: Tomara dos valores, “Lleno-Lleno” o “Costes adicionales”, la primera es que el coche reservado será entregado con el depósito lleno y debe ser entrega de igual manera, la segunda indica que tendrás que pagar un plus porque te lo entreguen con el depósito lleno, pero lo puedes devolver como quieras

politicaCancelacion:

Tipo: int

Descripción: Esta variable indica el precio que se cobrara en caso de que la reserva se cancele cuando queda menos de una semana

hrefFilial:

Tipo: String

Descripción: Uri del recurso filial asociado a este coche

modelo:

Tipo: String

Descripción: Modelo del coche

* Usuario:

Título: Usuario

Tipo: objeto

Descripción: Objeto que contiene los datos de un usuario

Requerido:

idUsuario

href

nombre

apellidos

correoElectronico

dni

direccion

ciudad

codigoPostal

fechaCarne

Atributos:

idUsuario:

Tipo: int

Descripción: Identificador único para cada usuario

href:

Tipo: String

Descripción: Uri de acceso al recurso dentro de la api

nombre:

Tipo: String

Descripción: Nombre del usuario

apellidos:

Tipo: String

Descripción: Apellidos del usuario

contrasena:

Tipo: String

Descripción: Contraseña puesta por el usuario

correoElectronico:

Tipo: String

Descipción: Correo electrónico registrado por el usuario

dni:

Tipo: String

Descripción: Cadena de nueve caracteres del documento nacional de identidad

direccion:

Tipo: String

Descripción: En esta variable se encontrará la dirección, calle y numero, o avenida, etc, del usuario

ciudad:

Tipo: String

Descripción: Ciudad en la que reside el usuario

codigoPostal:

Tipo: String

Descripción: Conjunto de cinco números indicando la zona en la que vives

fechaCarne:

Tipo: String

Descripción: Fecha en la que el usuario se sacó el carné de conducir

* Pago:

Título: Pago

Tipo: objeto

Descripción: Objeto que contiene los datos de pago de un usuario

Requerido:

idPago

href

titular

numTarjeta

hrefUsuarioPago

Atributos:

idPago:

Tipo: int

Descripción: Identificador de cada pago

href:

Tipo: String

Descripción: Uri de acceso al recurso dentro de la api

titular:

Tipo: String

Descripción: Nombre del propietario de la tarjeta

numTarjeta:

Tipo: String

Descripción: Cadena de 16 caracteres que indica el número de la tarjeta

hrefUsuarioPago:

Tipo: String

Descripción: Uri de acceso al usuario que hico el pago con esta tarjeta

* Reserva:

Título: Reserva

Tipo: objeto

Descripción: Objeto que contiene los datos de una reserva de un usuario

Requerido:

href

idReserva

fechaReserva

fechaRecogida

fechaEntrega

precioFinal

hrefFilialRecogida

hrefFilialEntrega

hrefCoche

hrefPago

Atributos:

idReserva:

Tipo: int

Descripción: Identificador de cada reserva

href:

Tipo: String

Descripción: Uri de acceso al recurso dentro de la api

fechaReserva:

Tipo: String

Descripción: Fecha en la que se hizo la reserva en formato YYYY-MM-DD

fechaRecogida:

Tipo: String

Descripción: Fecha en la que empieza el periodo de reserva en formato YYYY-MM-DD

fechaEntrega:

Tipo: String

Descripción: Fecha en la que termina el periodo de reserva en formato YYYY-MM-DD

precioFinal:

Tipo: float

Descripción: Precio total del tiempo reservado

hrefFilialRecogida:

Tipo: String

Descripción: Uri de acceso a la filial donde el cliente recogerá el coche

hrefFilialEntrega:

Tipo: String

Descripción: Uri de acceso a la filial donde el cliente entregara el coche

hrefCoche:

Tipo: String

Descripción: Uri de acceso al coche reservado

hrefPago:

Tipo: String

Descripción: Uri de acceso a los datos de pago de esta reserva

* Reservas:

Título: Reservas

Tipo: objeto

Descripción: Objeto que contiene listas de diez reservas como máximo, usado para devolver datos de diez en diez, proporcionando las uris de acceso a la siguiente decena de elementos, o a la previa.

Requerido:

href

reservas

Atributos:

href:

Tipo: String

Descripción: Indica la uri que has usado para acceder a estos datos

next:

Tipo: String

Descripción: Uri que apunta al siguiente grupo de datos

prev:

Tipo: String

Descripción: Uri que apunta al grupo de diez datos anteriores

reservas:

Tipo: Array

Descripción: Lista de diez objetos “reserva” máximo

Items:

$ref: Reserva

* CochesOfertados:

Título: CochesOfertados

Tipo: objeto

Descripción: Objeto que contiene listas de diez coches como máximo, usado para devolver datos de diez en diez, proporcionando las uris de acceso a la siguiente decena de elementos, o a la previa.

Requerido:

href

cochesOfertados

Atributos:

href:

Tipo: String

Descripción: Indica la uri que has usado para acceder a estos datos

next:

Tipo: String

Descripción: Uri que apunta al siguiente grupo de datos

prev:

Tipo: String

Descripción: Uri que apunta al grupo de diez datos anteriores

reservas:

Tipo: Array

Descripción: Lista de diez objetos “coche” máximo

Items:

$ref: CocheOfertado

* ReservaPago:

Título: ReservaPago

Tipo: objeto

Descripción: Objeto que contiene los datos de una reserva y un pago, se ha hecho de esta manera debido a que es más fácil recoger los datos cuando entran

Requerido:

href

cochesOfertados

Atributos:

reserva:

Tipo: objeto (Reserva)

Descripción: Contiene todos los datos necesarios de una reserva

pago:

Tipo: objeto (Pago)

Descripción: Contiene todos los datos del pago asociado a la reserva mencionada arriba

* 1. Métodos relacionados con el usuario.

Path: <http://localhost:8080/apirest/api/usuario>

Operaciones:

Método: POST

Descripción: Se le proporcionan los datos de un nuevo usuario para registrarlo en la base en caso de que no entre en conflicto con ninguno otro

Nickname: postUsuario

Consumes: application/json

Parámetros:

* + - Nombre: usuario

Descripción: Datos del nuevo usuario que se añadirá al sistema

Requerido: true

Tipo: objeto

Mensaje de respuesta:

Código: 400

Mensaje: Error: Correo electrónico ya registrado.

Código: 400

Mensaje: Error: DNI ya registrado.

Método: GET

Descripción: Este método comprueba las credenciales de un usuario y devuelve los datos del mismo

Nickname: comprobarUsuario

Produces: application/json

Parámetros:

* + - Nombre: dni

Descripción: Numero del documento de identidad

Requerido: true

Tipo: String

Tipo de parámetro: QueryParam

* + - Nombre: contrasena

Descripción: Contraseña puesta por el usuario al registrarse

Requerido: true

Tipo: String

Tipo de parámetro: QueryParam

Mensaje de respuesta:

Código: 404

Mensaje: Error: Usuario o contraseña invalido.

Código: Status 200

Path: [http://localhost:8080/apirest/api/usuario/{idUsuario}](http://localhost:8080/apirest/api/usuario/%7bidUsuario%7d)

Operaciones:

Método: GET

Descripción: Este método devuelve el coche cuyo id este en la uri, es un acceso directo a un recurso

Nickname: getUsuario

Produces: application/json

Parámetros:

* + - Nombre: idUsuario

Descripción: identificador de un usuario

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

Mensaje de respuesta:

Código: 404

Mensaje: Error: coche no encontrado

Código: 200

Path: [http://localhost:8080/apirest/api/usuario/{idUsuario}/pago](http://localhost:8080/apirest/api/usuario/%7bidUsuario%7d/pago)

Operaciones:

Método: GET

Descripción: Este método devuelve todos los pagos asociado al usuario cuyo id se encuentra en el Path, en caso de que no haya no devolverá nada

Nickname: getPagos

Produces: application/json

Parámetros:

* + - Nombre: idUsuario

Descripción: identificador de un usuario

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

Mensaje de respuesta:

Código: 200

Path: [http://localhost:8080/apirest/api/usuario/{idUsuario}/pago/{idPago}](http://localhost:8080/apirest/api/usuario/%7bidUsuario%7d/pago/%7bidPago%7d)

Operaciones:

Método: GET

Descripción: Este método devolverá los datos de pago cuyo identificador coincida con que se pasa en la uri

Nickname: getPago

Produces: application/json

Parámetros:

* + - Nombre: idUsuario

Descripción: identificador de un usuario

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

* + - Nombre: idPago

Descripción: identificador de un usuario

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

Mensaje de respuesta:

Código: 200

Código: 404

Mensaje: Error: pago no encontrado

Path: [http://localhost:8080/apirest/api/usuario/{idUsuario}/reserva](http://localhost:8080/apirest/api/usuario/%7bidUsuario%7d/reserva)

Operaciones:

Método: GET

Descripción: Este método devuelve todas las reservas de un usuario

Nickname: getReservas

Produces: application/json

Parámetros:

* + - Nombre: idUsuario

Descripción: identificador de un usuario

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

* + - Nombre: offset

Descripción: número a partir del cual se obtendrán los datos

Requerido: false

Tipo: String

Tipo de parámetro: Queryparam

Mensaje de respuesta:

Código: 200

Path: [http://localhost:8080/apirest/api/usuario/{idUsuario}/reserva/{idReserva}](http://localhost:8080/apirest/api/usuario/%7bidUsuario%7d/reserva/%7bidReserva%7d)

Operaciones:

Método: GET

Descripción: Este método devuelve los datos de una reserva en concreto

Nickname: getReserva

Produces: application/json

Parámetros:

* + - Nombre: idUsuario

Descripción: identificador de un usuario

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

* + - Nombre: idReserva

Descripción: identificador de una reserva

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

Mensaje de respuesta:

Código: 200

Código: 404

Mensaje: Error: reserva no encontrado

Método: DELETE

Descripción: Se comprueba primero que existe la reserva relativa al usuario indicado en la uri, tras comprobarlo se borra la disponibilidad relativa a la reserva, tras eso, la reserva en cuestión y, por último, si el pago solo existía debido a esta reserva, se eliminara también.

Nickname: deleteReserva

Parámetros:

* + - Nombre: idUsuario

Descripción: identificador de un usuario

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

* + - Nombre: idReserva

Descripción: identificador de una reserva

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

Mensaje de respuesta:

Código: 200

Código: 404

Mensaje: Error: No encontrado el pago

Código: 404

Mensaje: Error: No encontrado en reservas al borrar

Código: 404

Mensaje: Error: No encontrado en disponibilidad

Código: 404

Mensaje: Error: No se encuentra la reserva correspondiente al usuario registrado

Método: PUT

Descripción: El método actualizara la fecha de reserva en todas las tablas en las que se encuentra siempre y cuando la nueva fecha este disponible

Nickname: updateReserva

Consumes: application/json

Produces: application/json

Parámetros:

* + - Nombre: idUsuario

Descripción: identificador de un usuario

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

* + - Nombre: idReserva

Descripción: identificador de una reserva

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

* + - reservaNueva

Descripción: Datos nuevas de la reserva a actualizar

Requerido: true

Tipo: objeto

Mensaje de respuesta:

Código: 200

Código: 400

Mensaje: Error: reserva no actualizada

Código: 400

Mensaje: Error: disponibilidad no actualizada

Código: 400

Mensaje: Error: el nuevo periodo de reserva entra en conflicto con otro existente

Código: 400

Mensaje: Error: no se ha encontrado la disponibilidad asociada a esta reserva

Código: 404

Mensaje: Error: no se ha encontrado la reserva correspondiente a este usuario

Path: [http://localhost:8080/apirest/api/usuario/{idUsuario}/reserva](http://localhost:8080/apirest/api/usuario/%7bidUsuario%7d/reserva)

Operaciones:

Método: POST

Descripción: Este método consiste en crear una nueva reserva, añadiendo el nuevo periodo de indisponibilidad al coche y creando los datos de pago si no existen ya

Nickname: updateReserva

Consumes: application/json

Parámetros:

* + - Nombre: idUsuario

Descripción: identificador de un usuario

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

* + - Nombre: reservaPago

Descripción: Conjunto de datos de la nueva reserva y el pago que la ha realizado

Requerido: true

Tipo: String

Mensaje de respuesta:

Código: 201

Código: 400

Mensaje: Error: No coincide el titular con el numero de tarjeta

* 1. Métodos relacionados con la filial

Path: <http://localhost:8080/apirest/api/filial>

Operaciones:

Método: GET

Descripción: Este método devuelve todas las filiales que hay actualmente en el sistema

Nickname: getFiliales

Produces: application/json

Mensaje de respuesta:

Código: 200

Path: [http://localhost:8080/apirest/api/filial/{idFilial}](http://localhost:8080/apirest/api/filial/%7bidFilial%7d)

Operaciones:

Método: GET

Descripción: Este método devuelve los datos concretos de una filial

Nickname: getFilial

Produces: application/json

Parámetros:

* + - Nombre: idFilial

Descripción: identificador de una filial

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

Mensaje de respuesta:

Código: 404

Mensaje: Error: filial no encontrada

Código: 200

Path: [http://localhost:8080/apirest/api/filial/{idFilial}/oferta/{idOferta}](http://localhost:8080/apirest/api/filial/%7bidFilial%7d/oferta/%7bidOferta%7d)

Operaciones:

Método: GET

Descripción: Este método devuelve los datos de un coche ofertado indicado por el identificador

Nickname: getCocheOfertado

Produces: application/json

Parámetros:

* + - Nombre: idFilial

Descripción: identificador de una filial

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

* + - Nombre: idCoche

Descripción: identificador de un coche

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

Mensaje de respuesta:

Código: 404

Mensaje: Error: coche no encontrado

Código: 200

Path: [http://localhost:8080/apirest/api/filial/{idFilial}/oferta](http://localhost:8080/apirest/api/filial/%7bidFilial%7d/oferta)

Operaciones:

Método: GET

Descripción: Este método devuelve todas las ofertas de coches que coincidan con los criterios de búsqueda en grupos de 10 en diez

Nickname: getCochesOfertados

Produces: application/json

Parametros:

* + - Nombre: idFilial

Descripción: identificador de una filial

Requerido: true

Tipo: String

Tipo de parámetro: PathParam

* + - Nombre: fechaFilial

Descripción: fecha que indica el rango inferior de búsqueda

Requerido: true

Tipo: String

Tipo de parámetro: QueryParam

* + - Nombre: fechaFinal

Descripción: fecha que indica el rango superior de busqueda

Requerido: true

Tipo: String

Tipo de parámetro: QueryParam

* + - Nombre: offset

Descripción: indica a partir de que dato se tiene que enviar ahora

Requerido: false

Tipo: String

Tipo de parámetro: QueryParam

Mensaje de respuesta:

Código: 200