Prácticas de Sistemas Distribuidos

Creando una Agencia de Viajes

WS, RESTful, CRUD, MEAN, JSON

# Introducción

## Objetivo de la práctica

Identificar y definir los servicios necesarios para desarrollar la aplicación de **Agencia de Viajes** que se propone.

## Trabajo que realizar

La estudiante debe seguir las guías que se proporciona en este documento, realizando las siguientes tareas:

* Investigar acerca de los conceptos relacionados, fundamentalmente API y Arquitectura REST. Se presentará una breve memoria acerca de la investigación.
* Identificar, definir y describir los diferentes servicios que se precisarán. Se presentará una tabla resumen de los servicios identificados y una tabla por cada servicio en la que se den más detalles de cada uno de ellos.
* Por cada servicio, se diseñará su API siguiendo la plantilla que se adjunta en este documento. Se incluirá en la memoria una tabla resumen de los API y una ficha por cada end-point de cada API.

## Entregas

Se realizará una entrega a través de la aplicación **UACloud** en las fechas y formas que indicará el profesor de una memoria con todo el trabajo realizado.

La estructura de la **memoria** deberá contener, al menos, los siguientes elementos:

* Breve resumen o descripción de la investigación realizada sobre las API y arquitecturas REST
* Tabla resumen con los servicios identificados: nombre, tipo, descripción breve, etc.
* Figura con la arquitectura conceptual de dichos servicios.
* Tabla más detallada por cada servicio con los elementos que se consideren interesantes y una explicación más detallada sobre su cometido, funcionalidad, seguridad, etc.
* Tabla resumen de las API (tipo RESTful) que se crearán.
* Tabla expandida (según plantilla) de cada llamada al API.

# Recursos y referencias

Se trata de recursos interesantes para investigar y documentar toda la práctica. En este caso concreto se ha resaltado uno de ellos como el más adecuado para esta tarea específica.

Node.js

<https://juanda.gitbooks.io/webapps/content/api/>

NodeJS. Javascript en servidor

<https://juanda.gitbooks.io/webapps/content/javascript/node.html>

**API y Arquitectura REST (AJAX, CORS, Autenticación)**

<https://juanda.gitbooks.io/webapps/content/api/arquitectura-api-rest.html>

Creación de un API con NodeJS

<https://juanda.gitbooks.io/webapps/content/api/creacion_de_una_api_con_nodejs.html>

NodeJS y SSL/TLS

<https://www.sitepoint.com/how-to-use-ssltls-with-node-js/>

https://blog.mgechev.com/2014/02/19/create-https-tls-ssl-application-with-express-nodejs/

Registro, Autenticación y Autorización

<https://blog.restcase.com/4-most-used-rest-api-authentication-methods/>

JSON Web Token (JWT)

https://jwt.io/

https://www.npmjs.com/package/jwt-simple

https://www.npmjs.com/package/bcrypt

Seguridad adicional

https://expressjs.com/es/advanced/best-practice-security.html

Single Page Applications (SPA)

<https://juanda.gitbooks.io/webapps/content/spa/arquitectura_de_un_spa.html>

Evolución de CSS

<https://juanda.gitbooks.io/webapps/content/css/evolucion.html>

Proyecto de Web Básica

<https://juanda.gitbooks.io/webapps/content/proyectos/web_basica.html>

Web con Bootstrap y Gulp

<https://juanda.gitbooks.io/webapps/content/proyectos/bootstrap_gulp.html>

Desarrollo de aplicaciones Web

<https://www.gitbook.com/?utm_source=legacy&utm_medium=redirect&utm_campaign=close_legacy>

# Anexo 1. Plantillas y ejemplos de API RESTful

Esta guía se centra en la instalación, configuración y utilización básicas de las aplicaciones, *frameworks* y utilidades que necesitaremos para realizar la práctica.

Debe aportarse una tabla resumen en la que se puedan ver de un solo vistazo todos los servicios propuestos, algo del tipo…

| **Servicio** | **Tipo** | **Breve descripción** |
| --- | --- | --- |
| **Reserva de Coches** | WS tipo REST | Permite gestionar las reservas de coches. |
| **Reserva de …** | WS tipo REST | Permite gestionar las reservas de .... |
| **Registro** | WS tipo REST | Permite gestionar los usuarios del servicio |
| **Autenticación** | WS tipo REST | Proporciona un sistema de autenticación para los usuarios |
| **CRUD** | WS tipo REST | Conector con la DB |
| **Transacciones** | WS tipo REST | Gestiona las transacciones |
| **Pago** | WS tipo REST | Gestiona los pagos con las entidades bancarias y de crédito |
| **…** | … | … |

Son solo ejemplos de servicios. El estudiante debe incluir los servicios que estime conveniente para resolver la práctica. Igualmente, los campos de la tabla son sugerencias.

Para cada servicio se debe proponer un API. Dicho API se debería presentar de forma resumida en una tabla similar a la del ejemplo siguiente (correspondiente al ejercicio de API CRUD que hicimos en clase):

| **Verbo HTTP** | **Ruta** | **Descripción** |
| --- | --- | --- |
| **GET** | /api | Obtenemos todas las colecciones existentes en la DB. |
| **GET** | /api/{coleccion} | Obtenemos todos los elementos de la tabla {coleccion}. |
| **GET** | /api/{coleccion}/{id} | Obtenemos el elemento indicado en {id} de la tabla {coleccion}. |
| **POST** | /api/{coleccion} | Creamos un nuevo elemento en la tabla {coleccion}. |
| **PUT** | /api/{coleccion}/{id} | Modificamos el elemento {id} de la tabla {coleccion}. |
| **DELETE** | /api/{coleccion}/{id} | Eliminamos el elemento {id} de la tabla {coleccion}. |

Para cada uno de los servicios debería haber una tabla como la anterior y para cada uno de los end-points definidos en la tabla resumen de cada servicio, debería haber un desarrollo para el que se puede seguir la plantilla que se adjunta a continuación:

| **GET, POST, PUT, DELETE, …** | | **/api/xxx/{paramA}/yyy?paramB=abc** | Breve descripción del servicio o funcionalidad que ofrece el end-point |
| --- | --- | --- | --- |
| **Solicitud** (HTTP Request): | | | |
|  | **Cabeceras** (Headers):  Conten-Type: "application/json"  Authoritation: bearer <<ACCESS\_TOKEN>> | | |
|  | **Parámetros** (Parameters):  paramA: xxxxxx  paramB: xxxxx | | |
|  | **Cuerpo** (Body):  {  **abcd**: defg,  **ijklm**: hijk  } | | |
| **Respuestas** (HTTP Response): | | | |
|  | **Código de estado**:  200, 201, 400, 401, 500, … (suele haber más de una respuesta…) | | |
|  | **Cabeceras (**Headers**):**  Conten-Type: "application/json"  ... | | |
|  | **Cuerpo (**Body**):**  {  **result**: 'OK',  **yyyyy**: yyyy  } | | |

Generalmente hay una solicitud (HTTP Request), pero suele haber más de una respuesta (HTTP Response), al menos una para cuando todo ha ido bien y otra para cuando las cosas no han ido bien. Se deben definir todas las respuestas y se puede incluir una breve descripción de las mismas.