UNIVERSIDAD SAN PABLO - CEU

ESCUELA POLITÉCNICA SUPERIOR

GRADO EN INGENIERÍA DE SISTEMAS DE INFORMACIÓN



Diseño e Implementación de una

arquitectura RESTful para gestión el dia a dia de un estudiante de otro lugar

Design and Implementation of a

RESTful architecture for day-to-day management of a student from another place

Autor: David Recio Arnés

Tutor: Sergio Saugar García

Septiembre 2021

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Datos del alumno   |  | | --- | | Nombre:  David Recio Arnés |   Datos del Trabajo   |  | | --- | | TÍTULO DEL PROYECTO:  Diseño e Implementación de una arquitectura RESTful para gestión el dia a dia de un estudiante de otro lugar |   Tribunal calificador   |  |  | | --- | --- | | Presidente: | Fdo.: |  |  |  | | --- | --- | | Secretario: | Fdo.: |  |  |  | | --- | --- | | Vocal: | Fdo.: |  |  | | --- | | Reunido este tribunal el \_\_\_ /\_\_\_ /2021, acuerda otorgar al Trabajo Fin de Grado presentado por D./Dña. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ la calificación de \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |

Resumen

El objetivo de este trabajo de Fin de Grado es la creación de una aplicación portable (Servicios RESTFull) que se encargue de cubrir las necesidades de los estudiantes que llegan de otras ciudades a estudiar, elegí dicha temática por el creciente interés de los estudiantes, en estudiar en otros centros fuera de sus ciudades natales. Así, primero se realiza un exhaustivo análisis de las aplicaciones actuales que se encargan de cubrir las necesidades de los estudiantes y observando qué es lo que ofrecen y comparándolo con las necesidades reales de un grupo de personas. Mediante este análisis se puede llegar a la conclusión de que las aplicaciones sí cubren ciertos aspectos del día a día, pero no cumplen con todas las necesidades, ya que sólo se centran en una dinámica (estudio, planificación, herramientas) pero dejan de lado las demás necesidades sin lograr establecer un enlace entre estas.

Otro punto a tener en cuenta es la diversidad de elementos electrónicos que usan los estudiantes, por ello, la utilización de una arquitectura de Servicios Web RESTful como base para una aplicación de negocio es idónea, puesto que, una de sus ventajas asociadas es, precisamente, la capacidad de ser implementada en cualquier dispositivo con la mínima carga de implementación, ya que la mayor parte están en los servicios web (Back-end) y solo se debería crear el cliente para consumir estos (Front-end), permitiendo la integración desde una aplicación web, una aplicación de escritorio, etc.

Aunque principalmente esta aplicación está centrada en la implementación Android, no se descarta una futura implementación en iOS y en la creación de una aplicación web.

El objetivo final de este proyecto sería la creación de un prototipo de una aplicación que cubre una necesidad real y actual en el mercado, con la finalidad de subirlo a play store y hacer de este un negocio.

Palabras Clave

Aplicación, Back-end, Front-end, Estudiantes, Android, REST, servicios web.

Abstract

The objective of this End of Degree project is the creation of a portable application (RESTFull Services) that is in charge of covering the needs of students who come from other cities to study, I chose this subject due to the growing interest of students, in study at other centers outside their home cities. Thus, an exhaustive analysis of the current applications that cover the needs of the students is carried out first, observing what they offer and comparing it with the real needs of a group of people, through this analysis the conclusion can be reached that the applications do cover certain aspects of the day-to-day but do not meet all the needs, since they only focus on one dynamics (study, planning, tools) but leave aside the other needs without establishing a link between them.

Another point to keep in mind is the diversity of electronic elements used by students, therefore, the use of a RESTful Web Services architecture as the basis for a business application is ideal since one of its associated advantages is, precisely, the ability to be implemented on any device with the minimum implementation load, since most are in web services (Back-end) and only the client should be created to consume these (Front-end), allowing integration from a web application, a desktop application, etc.

Although mainly this application is focused on the Android implementation, a future implementation on iOS and the creation of a web application is not ruled out.

The final objective of this project would be to create a prototype of an application that covers a real and current need in the market, in order to upload it to the play store and make it a business.

Keywords

Application, Back-end, Front-end, Students, Android, REST, web services.

Índice de contenidos

Contenido

[Capítulo 1 Introducción 1](#_Toc52311790)

[1.1 Problema 2](#_Toc52311791)

[1.2 Objetivos 4](#_Toc52311792)

[1.3 Estructura de la memoria 5](#_Toc52311793)

[Capítulo 2 Gestión del proyecto 7](#_Toc52311794)

[2.1 Metodología 7](#_Toc52311795)

[2.2 Tareas realizadas 8](#_Toc52311796)

[2.3 Planificación 9](#_Toc52311797)

[2.4 Recursos 10](#_Toc52311798)

[Capítulo 3 Estudio del entorno 13](#_Toc52311799)

[3.1 Panorama actual 13](#_Toc52311800)

[3.2 Análisis del nicho 14](#_Toc52311801)

[3.3 Comparación entre las aplicaciones existentes 15](#_Toc52311802)

[3.4 Grupo de clientes 17](#_Toc52311803)

[3.5 Arquitectura 18](#_Toc52311804)

[3.5.1 SOA 18](#_Toc52311805)

[3.5.2 REST 26](#_Toc52311806)

[3.6 Tecnologías 35](#_Toc52311807)

[3.6.1 Front-end 35](#_Toc52311808)

[3.6.2 Back-end 42](#_Toc52311809)

[3.6.3 Bases de datos 44](#_Toc52311810)

[3.7 Resumen 48](#_Toc52311811)

[Capítulo 4 Análisis 50](#_Toc52311812)

[4.1 Apartado 3.1 51](#_Toc52311813)

[4.2 Apartado 3.2 51](#_Toc52311814)

[Capítulo 5 Diseño e implementación 53](#_Toc52311815)

[5.1 Apartado 4.1 54](#_Toc52311816)

[5.2 Apartado 4.2 54](#_Toc52311817)

[Capítulo 6 Evaluación 55](#_Toc52311818)

[6.1 Apartado 5.1 55](#_Toc52311819)

[6.2 Apartado 5.2 55](#_Toc52311820)

[6.3 Apartado 5.2 55](#_Toc52311821)

[Capítulo 7 Conclusiones y líneas futuras 57](#_Toc52311822)

[7.1 Apartado 6.1 57](#_Toc52311823)

[7.2 Apartado 6.2 57](#_Toc52311824)

[Capítulo 8 Conclusiones y líneas futuras 57](#_Toc52311825)

[8.1 Apartado 6.1 58](#_Toc52311826)

[8.2 Apartado 6.2 58](#_Toc52311827)

[Bibliografía 59](#_Toc52311828)

[Glosario de términos 61](#_Toc52311829)

[Anexos 63](#_Toc52311830)

* Metodología seguida.
* Enumeración y descripción de las tareas realizadas.
* Planificación seguida.
* Recursos utilizados.

Índice de ilustraciones

**No se encuentran elementos de tabla de ilustraciones.**

Índice de tablas

**No se encuentran elementos de tabla de ilustraciones.**

# Introducción

Los estudios internacionales (aquellas personas que eligen estudiar en el extranjero) siguen una tendencia creciente, tal y como se puede concluir de los datos proporcionados por la OCDE (ver gráfica 1) salvo un pequeño receso entre los años 2004–2008 debido a la crisis financiera de esa época.

Se estima que, en el futuro, este crecimiento lineal pasará a ser exponencial, debido al mundo globalizado en el que vivimos. Esto genera una serie de situaciones jamás antes vistas, tales como la permeabilidad de las fronteras entre los países de la UE (las cuales favorecen la entrada a los estudiantes de los países miembros) o la estandarización del lenguaje común dentro de la UE (el inglés). La consecuencia directa es que los estudiantes tienen un mayor abanico de posibilidades a la hora de escoger su centro educativo y, no sólo abarca el ámbito estudiantil, sino que también repercute sobre las empresas, siendo éstas últimas las que deben destacar para lograr captar el máximo de empleados talentosos posibles.

## Problema

Desde hace algunos años los problemas más importantes en el entorno universitario son la tasa de abandono y el fracaso académico.

Según un estudio realizado en el Colegio Oficial de la Psicología de Madrid (ver bibliografía), los puntos más importantes a tener en cuenta que determinan el éxito en una carrera y, en consecuencia, una tasa de abandono menor, hacen referencia a las variables tales como: variables comportamentales (hábitos de estudio, tales como control planificación, establecimiento de objetivos, etc.), variables afectivas (nivel de satisfacción), y variables motivacionales (factores internos y externos). Generalmente, la elección de una carrera por vocación tiene más probabilidad de éxito debido a un mayor rendimiento por parte del alumno, que una elección influenciada por el entorno, motivada más por presión de conseguir un mejor trabajo. Evidentemente, otro factor a tener en cuenta en cuanto a la tasa de abandono y, por ende, al fracaso académico, es la experiencia previa en los estudios; si el objeto de estudio tiene buenos hábitos (concentración, organización, bajo nivel de estrés en exámenes, etc.) los resultados serán más satisfactorios.

En cierta manera, esto puede verse mitigado utilizando un adecuado sistema de detección temprana del fracaso académico universitario, teniendo en cuenta las variables anteriores.

El abandono de los estudios universitarios supone el reconocimiento de un fracaso personal, debido a una falta motivacional como se vio en el estudio anterior, o una falta de aptitud para tales estudios.

En el CRUE 2017/2018 se plantea la siguiente hipótesis “*ceteris paribus*, los estudiantes con menores notas de acceso acabarán teniendo un peor rendimiento académico que, unido a la presión económica de las segundas matrículas, les dirigirán hacia el abandono”.

Tabla

Descripción generada automáticamente

*Tabla 1. Tasas de abandono por países. Fuente Education at a Glance 2019. OCDE y MCIU 2019. Informe CRUE Universidad Española en Cifras.*

Realizando una comparativa entre diferentes países, se puede extraer que el abandono escolar universitario es un hecho común en todos los sistemas universitarios, aunque no siempre afecta con el mismo grado. Si analizamos un poco más allá encontraremos una notable diferencia en la tasa de abandono de carreras universitarias de ciencias, tipo ingenierías, que en carreras de humanidades.

Imagen que contiene Tabla

Descripción generada automáticamente

*Tabla 2. Tasa de abandono en Universidades públicas presenciales españolas. Fuente: Informe CRUE Universidad Española en Cifras año 2017.*

Como ejemplo de la necesaria solución para mitigar el abandono universitario, el coste estimado del abandono global de los alumnos de la cohorte 2013/2014, matriculados en enseñanzas de grado ofertadas por el SUE, sería de 399,1 millones de euros.

Por otra parte, el fracaso académico no es homogéneo en cuanto a elección de carrera o universidad donde se imparte, por lo cual lo más conveniente sería adaptarlo al alumno en cuestión. Aun así, es bien conocido que los recursos de los cuales dispone en una universidad privada por lo general son superiores a los que dispone en una universidad pública, siendo partícipes en muchos casos de ese fracaso académico directamente.

En definitiva, podemos afirmar que el abandono universitario, principalmente durante el primer año, y el fracaso académico, suponen un problema real y de grandes repercusiones en el entorno educativo. Esto supone, además de un problema psicológico para el estudiante, una pérdida económica tanto para el alumno como para los países que subvencionan parte de los estudios, con lo cual es primordial encontrar una forma eficaz de solucionarlo.

Si extrapolamos estos problemas del entorno universitario al perfil de empleado que buscan las empresas, encontramos lo que denominan “habilidades blandas”. Lo que pretenden trasmitir hoy en día las universidades son conocimientos, pero eso dista mucho de las necesidades reales a la hora de encontrar un buen trabajo. Es muy importante y necesario hacer hincapié en reforzar las competencias transversales, incorporar a los conocimientos académicos otras habilidades y competencias, intensificar las prácticas en empresas y fomentar el emprendimiento y creación de empresas. Todo esto no es posible sin haber solucionado previamente el problema del que se parte al inicio de este estudio: tasa de abandono y fracaso académico.

## Objetivos

El objetivo principal de este proyecto es el desarrollo de una herramienta que sea capaz de ayudar al estudiante en la elección de una carrera universitaria (con un sistema basado en formularios, fácilmente comprensibles), en función de sus aptitudes y habilidades en distintas materias, planificación de los tiempos de estudio y mejorar el rendimiento orientado a su capacidad de concentración. Esto resolvería los principales problemas a los que se enfrentan los alumnos, sobre todo al estudiar fuera de su ciudad natal y disminuiría el fracaso académico y, por tanto, el abandono en los primeros años. Además, esta herramienta incluye un soporte que abarca la gestión económica diaria (mediante balances simples), con el fin de unificar los gastos derivados del estudio y de la convivencia del estudiante en su residencia, haciendo que sea lo más accesible posible para el individuo.

## Estructura de la memoria

En primer lugar, el capítulo 2 abordará la gestión realizada, detallando la metodología, planificación y recursos utilizados, continuando en el capítulo 3 con una exposición exhaustiva del entorno asociado al estudiante, analizando el panorama actual y la existencia previa de aplicaciones similares para tal fin. Estos puntos dan lugar a lo estudiado en el capítulo 4, donde se realiza un análisis de la aplicación, que contiene la especificación de los requisitos, un diagrama de flujo y una aclaración de resultados para que resulten más claros y concisos para el estudiante. Posteriormente, en el capítulo 5 se detalla el diseño e implementación de la aplicación, la cual será verificada en el capítulo 6 mediante la realización de un informe de pruebas con el objetivo de evaluarla. Por último, en el capítulo 7 se abordarán las conclusiones y los futuros evolutivos de dicha aplicación, cerrando así el ciclo de desarrollo de la misma.

# Gestión del proyecto

En este capítulo contendrá la planificación y metodología seguida para la elaboración del proyecto, para facilitar el entendimiento se aportarán diagramas, como en el caso del apartado 2.3.

## Metodología

La Metodología usada para el desarrollo de este proyecto será “Waterfall” o Cascada.

Esta metodología fue creada a principios de los 90 por Alan Davis y es considerada como una metodología clásica para el desarrollo de proyectos.

Es utilizada principalmente cuando los límites y funciones quedan establecidos y son firmes, es decir, que no debe cambiar durante el desarrollo de dicho proyecto.

A diferencia de XP (Extream programing) que este si admite cambios y es el más utilizado en estos tiempos debido a los constantes cambios a corto plazo que hay en los proyectos.

Estos cambios se ven bien marcados por las épocas y el avance de las tecnologías, donde los proyectos pasaron de una duración de desarrollo de 5 años o más a menos de un año, pudiéndose adaptar a las necesidades del mercado rápidamente.

El motivo principal por el que escogí la metodología cascada frente a metodologías agiles como xp, es que en xp se necesita un grupo de personas para hacer pear programing, siendo esto junto con las pruebas unitarias la base de esta metodología, imposible hacerlo en un trabajo de fin de grado, ya que no se puede más de una persona, por lo que tuve que descartarlo.

Teniendo en cuenta que este trabajo debe tener establecidos los límites y no va a estar en constante cambio, albergando una gran compatibilidad con la metodología cascada.

Para poder desarrollar una metodología en cascada serían las siguientes:

* Definición de requisitos
* Diseño de la arquitectura
* Implementación
* Testeo del programa e integración
* Mantenimiento

## 2.2 Tareas realizadas

En este punto se procederá a listar las diferentes tareas necesarias para el desarrollo del proyecto.

* Analizar a los actores y sus relaciones en el proyecto
* Realizar un Estudio del entorno donde se verá la situación actual en el mercado y el efecto que traerá esta aplicación en cuanto entre en el
* Investigar los diferentes patrones de diseño que puedan servir para agilizar la etapa del diseño de esta
* Realizar un diseño de la arquitectura siguiendo los principios REST, buscando siempre la simplicidad frente a la complejidad
* Creación de pruebas unitarias para las diferentes funciones de la aplicación
* Creación de pruebas para ver si posee todas las funcionalidades la aplicación
* Redacción de la memoria del proyecto.

## 2.3 Planificación

Para poder hacer este apartado más entendible decidí realizar un diagrama de GANT con todas las tareas especificadas y la cantidad de tiempo invertido en cada una de ellas.

En la parte izquierda de la ilustración se pueden observar las diferentes tareas que componen al proyecto, mientras que en la parte de la derecha se puede observar la duración de estas, quedando claramente visible su inicio y fin.

Se puede observar que la realización de la memoria es una tarea análoga a las otras ya que necesita de una constante a actualización.

ILUSTRACION 1 GANT

## 2.4 Recursos

Los instrumentos que se han utilizado en este proyecto serían los siguientes:

Sistema Operativo Windows:

Este es el sistema operativo que se ha utilizado para el desarrollo del proyecto, en el cual están instalados casi todas las herramientas utilizadas con ese mismo fin (Netbeans, GantProyect…).

Sistema Operativo Raspbian:

Sistema operativo de pruebas. Escogí este sistema operativo ya que hace no mucho adquirí una raspberry pi4 y elegí trabajar sobre ella ya que se encuentra en un entorno aislado y de fácil acceso en el cual puedo aprender mientras desarrollo diversos proyectos en ella.

GantProyect:

Herramienta utilizada para la planificación de tiempos dedicados a las tareas, es de uso gratuito. El motivo fundamental de su uso fue que ya tenía esperanza en ella, dado que aprendí a usarla en programación 1

GitHub:

Repositorio bastante conocido utilizado para mantener una estructura correcta en las diferentes versiones del proyecto, pudiendo ver los avances sin temor a corromper el código, debido, a que si se sigue una correcta gestión del repositorio puedo volver a un punto cercano sin mucha perdida de trabajo.

NetBeans

Herramienta utilizada para el desarrollo del código de los servicios web que consumirá la aplicación, para poder seguir la filosofía REST, haciendo así que la aplicación sea más liviana y portable.

MySQL:

Herramienta utilizada para la gestión de los datos a nivel de prueba antes de ser subida a la raspberry, siendo esta ultima el entorno final ya que se espera en un futuro alojar los servicios web en un servidor aparte con Linux

AndroidStudio:

Herramienta de desarrollo para la creación de aplicaciones móviles, bastante reconocida y posee una interfaz que simula un móvil con el sistema operativo que quieras y el modelo de Android que quieras.

Paquete de Office

utilizado para el desarrollo de la documentación o información complementaria a estos

Apache:

Herramienta utilizada para la publicación de servicios web que serán consumidos por la aplicación

Putty:

Herramienta de conexión de forma remota, la cual tiene asociada la ip publicad de la maquina destino

MariaDB:

Herramienta similar a MySQL instalada por temas de compatibilidad en la raspberry con la función de alcanzar los datos necesarios en la aplicación

# Estudio del entorno

En este capítulo, se abordarán todos los eventos del mercado con los que se encontrara dicha aplicación y su efecto sobre esta. Para ello se proporcionará una vista al panorama actual, comparación entre aplicaciones existentes, viabilidad de la aplicación, análisis del nicho de mercado sin explotar, toma en contacto con un grupo de personas que poseen estas necesidades, formas de arquitectura más utilizadas y por último el resumen de los puntos clave.

## Panorama actual

La situación actual se encuentra muy convulsa, los mercados no poseen en su mayoría una estabilidad mayor a un año, fácilmente comparable a riadas, es decir, son corrientes fuertes y repentinas con muchos cambios, por tanto, las empresas luchan cada día por destacar y aprovecharse de las fuertes corrientes, para ello, la flexibilidad y la portabilidad son dos elementos totalmente necesarios y vitales.

Durante estos últimos años se está produciendo una migración transitoria hacia la nube virtual, ya sea por su utilidad como almacén de datos o como lugar de almacenamiento de programas, entre otros. Esto se pudo llegar a conocer en fuentes respetables como **“EuropaPress”** que lanza el siguiente titular “***la inversión en cloud se triplicará en España hasta 2023, cuando alcanzará los 2.800 millones, según IDC”*** [[1]](#footnote-1)***.*** Mostrando claramente el avance de estas tecnologías.

Por otro lado, la cantidad de personas que usan los diferentes aparatos móviles ha ido creciendo, haciendo imprescindible estos aparatos tanto para la vida laboral como individual.

## Análisis del nicho

Teniendo los puntos anteriores, se puede empezar a intuir la base de este proyecto, ahora lo siguiente que se debe hacer es analizar el nicho que se va a explotar, en este nicho el principal cliente es aquella persona que pretende formar se en zonas alejadas de su lugar de origen, principalmente jóvenes adultos.

Llegados a este punto se dividirá las necesidades en dos cauces:

* Jóvenes adultos:

Los jóvenes adultos se encuentran en una etapa de la vida bastante difícil (fin del bachiller e ingreso en un grado o formación superior).

En esta etapa los primeros años son decisivos, ya que se enfrenta a asignaturas más complejas que en la escuela y muchas de ellas no tiene las bases ya estudiadas, a esto hay que sumarle el coctel de hormonas que les dicen que ya son mayores para ciertas cosas y para otras son pequeños generando al individuo un estrés innecesario, reflejado en muchos casos como abandono de los estudios.

Para apalear todas estas dificultades el estudiante se sirve de aplicaciones tanto con interés lectivo, como social.

* Personas alejadas de su lugar de origen:

En este caso nos movemos en otro ambiente, pero no más sencillo, dado que la persona se encontrará sola en una zona que apenas conoce, en muchos casos no tiene la capacidad de autogestionarse correctamente, es decir, no sabe planchar, cocinar, fregar, etc.…

También no sabe de la ubicación de centros comerciales donde pueda adquirir los recursos para cubrir sus necesidades.

## Comparación entre las aplicaciones existentes

Tras una primera aproximación al mercado, pude ver que hay un conjunto de aplicaciones bastante extenso, pero ninguna agrupa las funcionalidades ni las relaciona, dejando un nicho viable para esta idea. Estas aplicaciones pueden ser:

My Study Life

Que posee una idea bastante llamativa. Organizar el día a día según un horario de estudio que tu configuras, un apartado que explica las asignaturas y el profesor, tareas asociadas, posibilidad de introducir exámenes

Exam Countdown

Cuenta regresiva y planificador ante un examen ya establecido con la posibilidad de establecer un plan de ataque hasta que suceda el evento(examen)

Studious

Es similar a My Study Life, pero con la opción de silenciar el teléfono o activarse solo con las tareas programadas o permitidas que funciones

Brain Focus

Enfocada más a un horario a cada tarea que tu planificas estableciendo un horario de comienzo de la tarea y un horario de finalización de la tarea

Analizando en profundidad estos conceptos de ideas se puede establecer que son servicios interesantes en este estado, es decir, de forma individual poseen un gran poder, pero les falta un director de orquesta para subirlos a un mayor nivel.

Por ejemplo, si tomamos la idea principal de “My Study Life” podríamos ofrecer un servicio dedicado al horario escolar donde se le podrían poner las diferentes tareas que tienen las asignaturas del día, ahora bien, le falta una vuelta de tuerca, ya que aparte de saber que tareas tienes en el día, necesitas saber cuáles son más prioritarias, no es lo mismo una tarea con un tiempo de 2 semanas que una de 2 días de la misma asignatura. Pero no es solo una vuelta de tuerca, lo más importante de esto es dividir las funcionalidades de las aplicaciones existentes, retocarlas buscando una mejora sin perder la esencia, y unirlas a otras.

Persiguiendo este concepto se puede llegar a cosas tales como lo anterior (horario con tareas jerárquicas) y le proporcionamos la funcionalidad de “Studious”, poniendo alarmas a las tareas, silenciándote el teléfono para evitar distracciones y al final de un tiempo determinado realizando otro sonido, indicativo que dejes la tarea.

## Grupo de clientes

Respecto a este proyecto tuve la suerte de conocer un conjunto de personas no solo a nivel nacional, sino a nivel internacional que me expresaron la cantidad de dificultades que tienen que hacer frente para llevar el día a día en esta situación y sus inquietudes inherentes a las mismas,

En este caso establecí una gran división entre ellos, más que nada por su cultura para poder analizarlos en mayor profundidad.

Por un lado, tengo estudiantes que residen en México, los cuales optaron o bien por un intercambio a otro país, o bien se mudaron a otro estado.

Un dato que me pareció curioso es la analogía tan clara respecto a unos problemas tan tribales, que en un principio quería descartarlo, pero al ver que se repetía tanto opte por incluirlo en el proyecto, este problema sería la gestión de tareas de la casa junto con el estudio. Siempre andaban diciendo que es muy difícil realizar tareas de la casa sobre todo al principio donde no posee experiencia, y que les gustaría tener una ayuda sea o una guía o un foro donde puedan aprender técnicas que le faciliten este problema, claro está, sin dejar de lado el tiempo dedicado a los estudios. Otro factor que les pesaba en muchos casos es el factor del trabajo ya que muchos de ellos trabajan a tiempo parcial, eso suponía un ajuste mayor en el tiempo, donde pecaban de optimistas, hasta claro está el momento del examen donde sufrían las consecuencias de esta mala gestión del tiempo.

Por otro lado, tengo al grupo de nacionalidad española, donde si es cierto que también poseen trabajo a tiempo parcial en algunos casos, no es excesivamente común, sobre todo si tienen una gran carga de estudios.

## Arquitectura

Para poder establecer un camino en el desarrollo de un proyecto, es necesario realizar un plan lo suficientemente elaborado para que este no se derrumbe fácilmente y es en este punto donde la arquitectura tiene un papel clave para establecerlo.

Para entender mejor este “plan” a continuación se darán unas pinceladas concisas de los conceptos de esta. Pero antes debemos empezar por la precursora SOA para luego poder hablar de la filosofía RestFull.

### SOA

La arquitectura SOA se remonta a 1996 descrita por Gathner, pero no fue hasta 2010 donde empezó el auge de los servicios web. Siendo esta la arquitectura más socialmente aceptada por la comunidad, pero cabe destacar que SOA no es inherente a los servicios web.

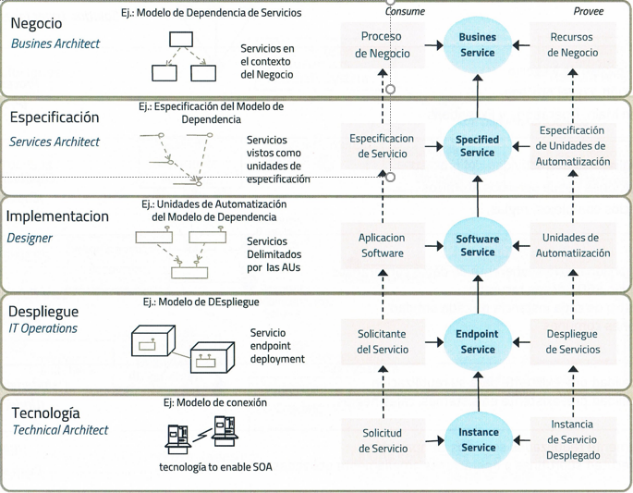
Para entenderlo mejor, será separado en los siguientes puntos:

* SOA, Fundamentos, Niveles de Abstracción y Modelo Metodológico
* Técnicas para el diseño de Modelos Ontológico y de Dominios
* Técnicas para el diseño de CoreBiz Services Layer
* Técnicas para el diseño de Process Services Layer
* Técnicas para el diseño de Capability Services Layer
* Utility, Underlying y otras posibles capas de servicios
* Ejemplos de Modelización de una SOA

#### **SOA, Fundamentos, Arquitectura de servicios y Niveles de Abstracción**

**La palabra clave** para entender todo es servicio. Un servicio es un conjunto de actividades o funciones que buscan responder a las necesidades de un cliente, o también dicho, un servicio es la unidad mínima y necesaria para responder a las necesidades del cliente.

**SOA** trata de identificar los servicios core y los encapsula de forma estandarizada para poder hacerlos más atómicos y reutilizables (como un lego).

**Una arquitectura de servicios** es una colección de Servicios, clasificada en tipo, organizada en capas, gobernada por patrones y políticas, gestionada por un Centro de Gobierno

Negocio: propiamente el negocio

Especificación: cada servicio que responde

a una actividad del negocio

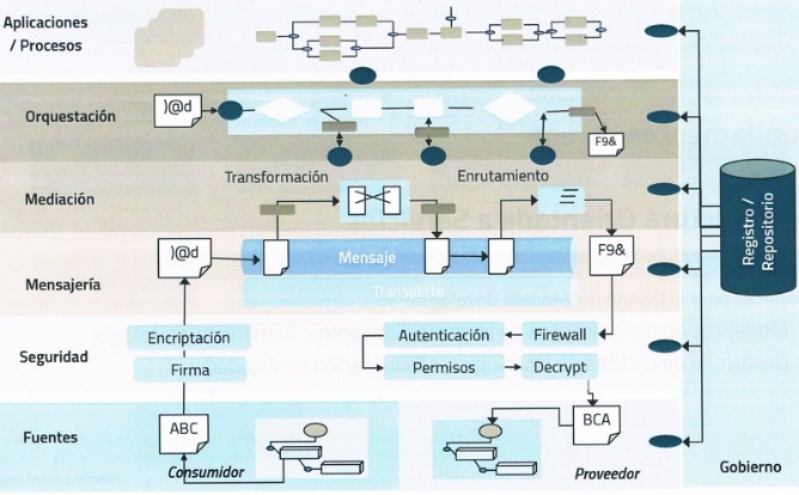
Implementación: relaciones entre servicios

creando así un servicio más complejo

Despliegue: puesta en marcha de los servicios

Tecnología: son las máquinas necesarias

**Los niveles de abstracción** serían los siguientes:

Aplicaciones: son las aplicaciones del negocio

Orquestación: es el director que pone en marcha

la sincronía de las capas

Mediación: donde se valida el paso de información

Mensajería: donde se mueven los ficheros XML

Seguridad: donde se encripta y desencripta

la información

Las fuentes: se extrae la información necesaria

Registro: Alancen de datos

#### **Técnicas para el diseño de Modelos Ontológico y de Dominios Técnicas para el diseño de Modelos Ontológico y de Dominios**

#### **Técnicas para el diseño de Modelo Ontológico**

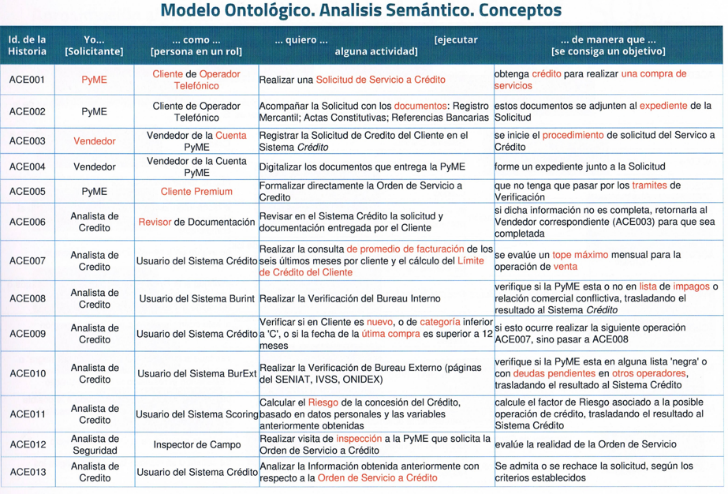
La ontología es la descripción explicita y formal de las clases. Para desarrollarla correctamente se deben seguir las siguientes fases:

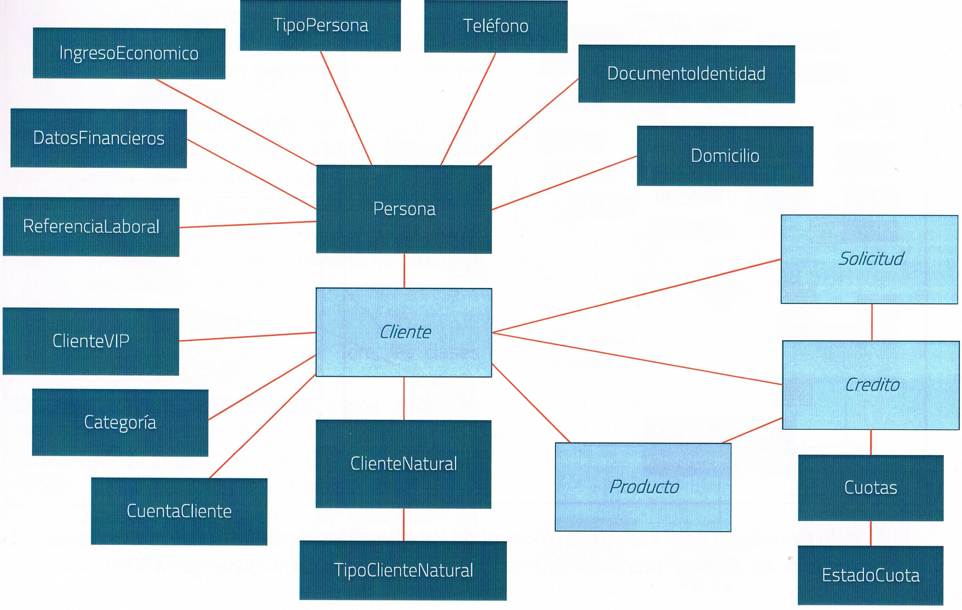
1. Definición del Dominio y el Alcance
2. Reutilización de Ontologías
3. Enumerar Términos Relevantes
4. Definición de las Clases y la Jerarquía de Clases
5. Definición de Roles
6. Definición de las Restricciones de Rol
7. Creación de Instancias

Para entender mejor estas fases, se debe explicar la siguiente lista de conceptos:

* CLASES: Describen conceptos
* SUBCLASES: Representan Conceptos específicos
* INSTANCIAS: Individuos específicos de una clase
* ROLES: Describen propiedades de las clases e instancias

Ejemplo de caso de negocio, el cual se analizará y se obtendrá un diagrama de clase y funciones



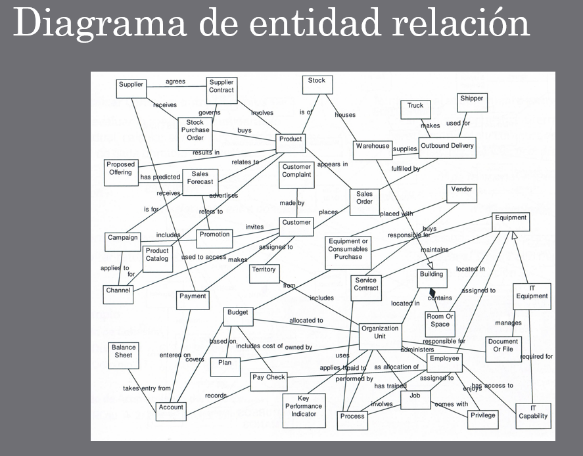


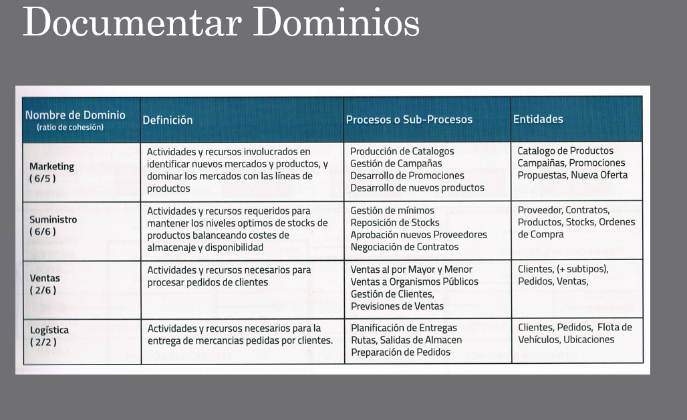
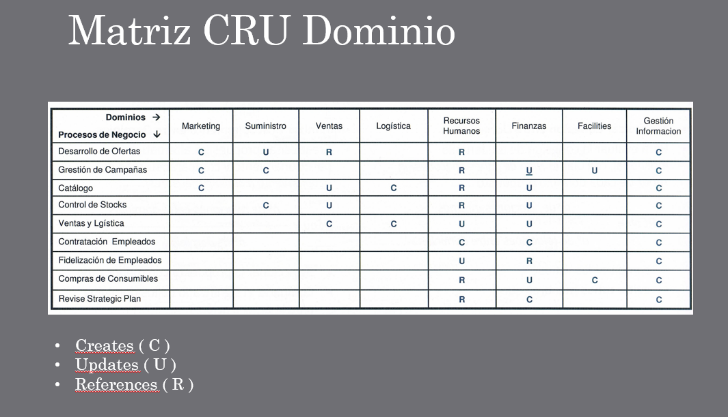
#### **Técnicas para el diseño de Dominios**

**Un dominio SOA** son particiones lógicas de una empresa en un conjunto de recursos al subdividir la organización en Dominios nos permite la planificación de sus actividades. Los Servicios de un dominio pueden compartir políticas y semántica.

**La técnica para identificar los dominios** se consigue siguiendo una serie de pasos:

1. Comenzar con un diagrama de entidad relación del alcance
2. Sobre el diagrama dibujar líneas delimitando las fronteras de dominio
3. Agrupa los dominios (por funcionalidad) optimizando la Cohesión y Acoplamiento
4. Matriz CRU Dominio, cómo se ven afectados por los procesos del negocio
5. Documentar Dominios

Ejemplo



#### **Técnicas para el diseño de CoreBiz Services Layer**

Los CoreBiz Services son el centro de los recursos TI de la Organización, los cuales guardan registros de cada instancia de cada entidad de que son responsables. Solo poseen único punto de operación sobre ellos, dado que son servicios de alto nivel, capaces de proporcionar una visión de 360º, es decir, Es el corazón del negocio, con él te puedes hacer a la idea de la estructura de la empresa y siempre son de alto nivel.

Cuando la complejidad es excesiva en esta capa, se divide en:

* Reference Business: operaciones de consulta
* CUD Business: solo operaciones CUD

Para identificarlos debes seguir los siguientes pasos:

1. Prerrequisitos
2. Determinar los identificadores en el diagrama entidad relación
3. Identificar Entidades del Núcleo
4. Identificar y asignar Entidades de Detalle y Entidades de Clasificación
5. Nombrar y revisar los ‘Core Business Services’
6. Preparar Descripciones Iniciales de Servicios
7. Identificar Dependencias entre Servicios
8. Preparar Diagrama de Dependencias de Servicios

#### **Técnicas para el diseño de Process Services Layer**

**Los Process Services** son los encargados de Secuenciar operaciones de varios servicios (Humano->maquina) como de Secuenciar las operaciones necesarias para incorporar las actividades del usuario (Maquina->Maquina).

Para identificarlos se debe tener en cuenta lo siguiente:

* Un Process Service por cada Proceso de Negocio
* Identificar las funciones necesarias
* Cada función representa una dependencia
* La Función puede ser cualquier tipo de Servicio

A su vez la capa de Process Services soporta la estructura del diseño de Gestión de Procesos Automatización:

* Main Process Layer
* Operations Process Layer
* Exceptions Procees Layer

Cabe destacar que una parte importante es la subdivisión de los servicios de esta capa para obtener servicios más atómicos, para ello tenemos que dividir los grandes Process Services en varios servicios asignados a subprocesos, luego buscar los subprocesos comunes que puedan ser compartidos desde varios Procesos de Negocio, estructurarlos y se obtendrá como resultado una disminución del peso del Proceso a la hora de ser ejecutado.

#### **Técnicas para el diseño de Capability Services Layer**

Una empresa puede considerarse como una plataforma de capacidades, donde una capability es un elemento estable de una operación y estas deberían sobrevivir a los Procesos.

La técnica para identificarlo seria mediante la descomposición Funcional, se determinan cuáles son las Actividades en las que se descompone un proceso. Cada actividad a su vez puede ser descompuesta en una serie de acciones que están soportadas por funcionalidad de sistemas y que el actor utiliza para completar la actividad. Cada una de dichas acciones identifica a un Activity-Function Service (AFS), que se representará en la capa Activity-Function Layer, luego los AFS sustituyen a los legacy con ayuda de servicios que le dan las funcionalidades necesarias. Por último, se retocan los AFS para que pasen ser core sin necesidad de dichos servicios.

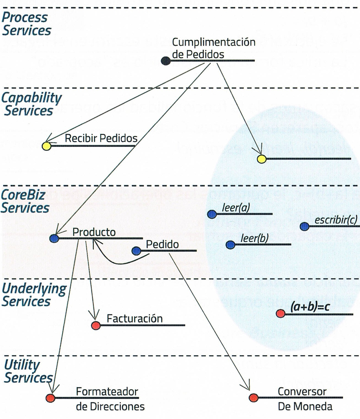
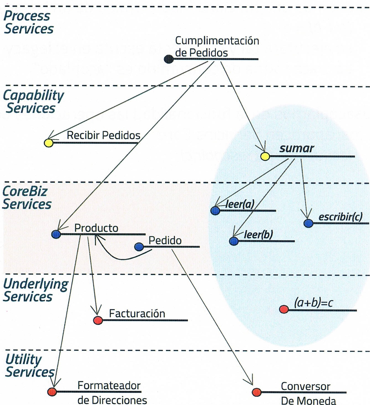
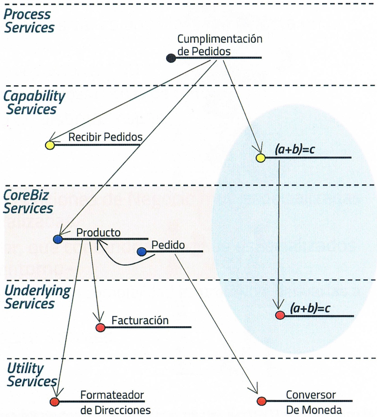
#### **Utility, Underlying y otras posibles capas de servicio**

Utility: Realizan rutinas comunes, necesarias a varios servicios situados en distintos dominios. Suelen ser servicios orientados al negocio, los cuales pueden suministrar la infraestructura técnica, pero siempre son reutilizables y compartidos.

Underlying: Aquellos servicios difíciles de migrar, ocultados en cajas negras hasta que se logran migrar y están íntimamente relacionados con los Sistemas Legados

Capability: Funciones de Negocio muy especializadas

#### **Ejemplo de Modelización de una SOA**



### REST

Antes de nada, cabe destacar que REST en si no es una arquitectura, sino es un estilo de arquitectura.

Cuando quieres aplicar REST en una aplicación web, tienes que imaginarte que la aplicación en si es una máquina de estados, donde los distintos nodos son los recursos y los enlaces presentan una transición de estado, por tanto, el siguiente estado es el resultado de seguir el enlace y el recurso que se acceda.

#### **Propiedades de REST**

Para poder entender mejor la filosofía REST, se expondrán sus propiedades para una mejor comprensión:

* Heterogéneo

Hace referencia que es un conjunto de funciones, usadas para diferentes interacciones como los delete, put, post, etc

* Escalable

Al ser más flexible que las otras arquitecturas, esta filosofía permite mediante una modificación más sencilla acoplarle diferentes funcionalidades según avanza su desarrollo.

* Evolutivo

Al ser flexible se adapta a cualquier cambio, siendo capaz de superar cada obstáculo, por ejemplo, que se rompa el móvil, al estar el servicio en la red, este puede ser perfectamente consumible.

* Visible

Significa que es alcanzable de forma sencilla en la red para una mayor accesibilidad a los recursos publicitados.

* Fiable

Ofrece la posibilidad de cifrar la seguridad de la información que se intercambia entre el cliente y el servicio que se publicita, por ejemplo, mediante tokens

* Eficiente

Al dividir los tipos de interacciones en las funciones permite que los accesos sean diferentes en pesos, por ejemplo, no es lo mismo un pos donde realiza un cambio esperando una respuesta obligatoriamente, que un delete, donde se borran datos.

#### **Restricciones de REST**

REST al no tener realmente unas pautas y puntos a seguir no se considera como una arquitectura, pero si posee una serie de restricciones que hace que cualquier arquitectura que quiera seguir REST será modelada por las limitaciones.

Estas restricciones son:

* Servidor cliente

Esto significa que el cliente debe estar separado del servidor y ambos deben estar aislados, de esta manera se puede ofrecer el mismo servicio a diversos clientes con una abstracción mayor, a su vez hace que los cambios son más fáciles de aplicar ya que son cosas diferentes

* Apátrida

Esto hace referencia a que cada servicio ofrecido por el api REST debe ser completo, es decir, la solicitud enviada debe tener lo necesario para el servicio que es atacado, y este debe dar una respuesta coherente.

* Caché

Un servicio de api REST debe ser rápida y eficiente, para ello debe tener una cache para agilizar las respuestas, pero, el mismo usuario debe tener la capacidad de borrarlo

* Interfaz uniforme

La definición de los recursos consta de:

* + URI
  + Formato de intercambio de datos (XML)
  + Métodos (GET, PUT, Post.)

Los recursos se identifican de forma unívoca por su URI, estos servicios tienen la capacidad de leer, crear, modificar, y eliminar recursos

* Sistemas de capas

Al tener este sistema, la aplicación adquiere una estabilidad muy grande, ya que permite una jerarquización de estas, pudiendo encapsular las funcionalidades en cada una de las mismas, fomentando la escalabilidad.

* Código bajo demanda

Estamos en una época cambiante y esta tecnología ofrece la posibilidad de adaptar el código a los diferentes obstáculos que se enfrenta en su periodo de desarrollo de la aplicación.

#### **Ventajas de REST vs SOAP**

Para poder compararlos hay que tener claro que no se pude hacer de manera directa ya que SOAP es un protocolo, mientras que REST es un estilo arquitectónico, también hay que tener en cuenta que hay muchas personas tienen tendencias a llamar a lo que no es SOAP lo llaman REST, para intentar solucionarlo se crearon los niveles de REST.

Para entenderlo mejor, se expondrá las cualidades de ambos en una tabla.



#### **Servicios de RestFull**

Leonard Richardson es un experto en RestFull dejo los siguientes conceptos entre muchos otros, en los libros que escribió.

Como cualquier otra tecnología, La tecnología web no soluciona automáticamente los problemas de las aplicaciones de negocio o los problemas de integración, pero unas venas prácticas de diseño, un buen testeo y un extenso número de patrones usados, puede tomar una gran cantidad de tiempo para crear un buen servicio web.

Leonard Richardson fue el que propuso una clasificación de los servicios web, esta clasificación se divide en cuatro niveles:

* Nivel 0

Es el nivel más básico, donde los servicios que tienen una única URI, a su vez solo utilizan un único método HTTP (POST típicamente) pero claro, sin aprovechar las ventajas y las opciones que da (“tunelling” using HTTP)

Un ejemplo claro de este nivel sería un servicio SOAP.

* Nivel 1

Es en este nivel donde se quedan la mayoría de los servicios REST, ya que no supone un gran avance respecto al anterior.

En este nivel se exponen varios recursos, esto se puede dar gracias a que se pueden usar más de una Uri, pero únicamente un método HTTP (normalmente GET o POST).

* Nivel 2

Es en este nivel donde se utilizan varias URIs que además soportan varios métodos HTTP, además se aprovecharan las ventajas que ofrece HTTP, tales como, los códigos de respuesta, dejando de utilizar HTTP solo para trasporte como en los anteriores niveles. Aparte ya en este nivel tenemos servicios que soportan operaciones CRUD

* Nivel 3

En este nivel se añade la noción de HATEOAS, es decir, los recursos tienen enlaces que apuntan a su vez a otros recursos, donde el cliente mantiene el estado de la aplicación, mientras que la aplicación sabe tras cada respuesta cuales son las posibles continuaciones.

#### **Buenas prácticas en REST**

Para desarrollar una buena APIREST y que esta sea entendible y simple se deben seguir una serie de puntos:

* A la hora de definir un servicio, su paht deben ser sustantivos, nunca verbos, ya que esto dificultaría la comprensión de forma innecesaria.
* Tener múltiples URIs(paht) para diferenciar el acceso a cada recurso.
* Las asociaciones deben ser simples y concretas, buscando siempre que la API sea muy intuitiva
* No se debe sobrecargar al cliente, es importante saber que el cliente no siempre requiere toda la información y por tanto debe ser filtrada y en caso de ser un resultado muy grande usar la paginación
* Hay que tener en cuenta que las peticiones GET tienen un tamaño delimitado, por tanto, si necesitas introducir muchos argumentos o con valores de tamaño variable te puedes servir de peticiones como POST o PUT.
* Es importante utilizar los estándares como los códigos de error de HTTP y utilizar correctamente los métodos haciendo que sea más intuitiva
* Para buscar, ordenar, filtrar, paginar, etc No se crearán nuevos recursos, sino que se añadirán los parámetros al GET simplificando el código y quitándole carga al mismo.
* Hay que avisar al cliente durante las actualizaciones si va a haber compatibilidad con versiones anteriores, manteniendo el nombre, o cambiando el nombre, indicando que no hay compatibilidad hacia atrás.
* Usar un mapeo de URIs lo más intuitivo posible y no cambiarlo para evitar confusiones
* Hay que pensar muy bien cómo va a realizar el consumidor todas las operaciones necesarias puesto que debe ser intuitivas para un uso más fácil.
* Aplicar HATEOAS.

#### **REST en Java**

En este proyecto se decidió desarrollar REST en java, por tanto, hay que definir el cómo se puede emplear en este lenguaje y cuál es su objetivo.

* Especificaciones JAX RS 1 1 y JAX RS 2 0

Para utilizar Res es necesario que tengamos el soporte necesario(librerías) para poder desarrollar el servidor de aplicaciones al cual le mandaran las peticiones.

* Jersey

Es el framework que será utilizado para simplificar la implementación y despliegue de los servicios REST.

Las ventajas que ofrece este software,

* + Un software maduro utilizado en entornos de desarrollo, esto significa que posee una gran cantidad de guías para las dudas que pueda tenerse durante el desarrollo del proyecto.
  + Es open Source, por lo tanto, no supondrá ningún coste el emplearlo.
  + Permite utilizar filtros para realizar tareas fuera del ámbito de la lógica REST, como, por ejemplo, realizar cambios en las peticiones HTTP antes y/o después de ejecutar el código del servicio REST.
  + Permite la definición de servicios mediante métodos de clases Java con anotaciones JAX-RS. Estas anotaciones pueden ser:
    - @Path -->URL del servicio
    - @GET @PUT @POST @DELETE definen el método HTTP
    - @Produces define el tipo de respuesta del servicio
    - @Consumes define el tipo de entrada que espera Ej. text plain
    - Context Nos permite hacer referencia a recursos, cabeceras, etc
    - PathParam Parámetro que se obtiene de la URI directamente
    - QueryParam Parámetro que se envía junto con la URI

## Tecnologías

En este apartado se procederá a exponer las diversas tecnologías implicadas en el desarrollo del proyecto. Dado la temática del proyecto, este apartado se dividirá en dos partes: front-end siendo esta parte la encargada de recoger las interacciones del usuario con la aplicación y es esta parte la que mandara los datos obtenidos de esas interacciones al back-end. Esta segunda capa no la percibe el usuario, la cual es la encargada de tratar los datos recibidos mediante sus funciones internas y ofrecer una respuesta que será enviada al cliente.

Después de describir estas dos partes, se definirá las bases de datos utilizadas y el motivo de su uso.

### Front-end

Como ya se mención anteriormente, esta capa es la encargada de la parte del usuario/cliente.

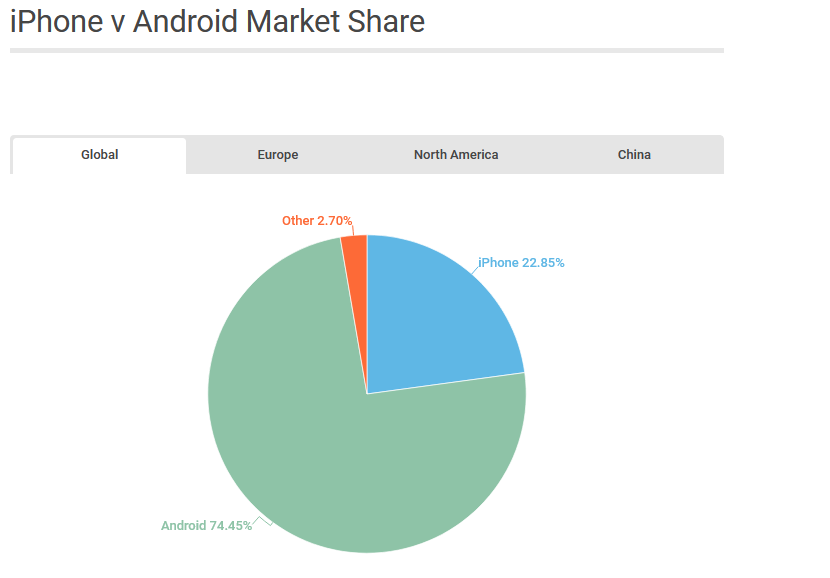
Para esta parte se expondrán el por que se usan estas tecnologías, y una descripción de estas.

#### **Android**

Esta tecnología es necesaria para poder realizar la entrada al mundo móvil, permitiendo una gran portabilidad de esta. Como ya se expuso en el apartado 3.1, el móvil ya forma parte esencial en nuestras vidas, por lo que es imprescindible que haga esta transición.

Una vez analizado el porqué, se procederá a explicar la tecnología.

Android es un sistema operativo que compite de forma directa con IOS siendo estos los lideres del mercado, aunque actualmente la mayoría usa Android.

[[2]](#footnote-2)

Una cosa curiosa de este sistema operativo es que fue desarrollado por Google y se basa en el kernel de Linux, además este sistema operativo facilita el uso de diversas aplicaciones, tanto oficiales como no oficiales (creadas por ti).

Para entenderlo mas en detalle a continuación se mostrará la historia de este sistema operativo.

La historia comienza con una empresa llamada Android Inc fundada en 2003, fundada por Andrey Rubín, Rich Miner, Nick Sears y Chirs White. En un principio esta empresa desarrollo este sistema operativo orientado a las cámaras digitales, permitiéndolas conectarlas al pc sin cables, pero al ver que no tenia mucho futura, decidieron introducirse en el mundo de los teléfonos móviles. Pasados dos años la empresa es comprada por Google, ya que este siempre que ve una empresa emprendedora y rompedora en el área tecnológico suele comprarla.

En 2007 se anuncia la primera versión del sistema operativo de Android en móviles, esta se llamó “Android 1.0 Apple Pie”, junto la creación de este sistema operativo se creó la Open Handset Alliance[[3]](#footnote-3), lo que hizo que Google liberase la mayor parte del código de Android bajo licencia de Apache, es decir, de código abierto y licencia libre.

Habrá que esperar hasta mediados de 2008 donde aparecieron los primeros dispositivos con esta versión de Android.

Durante el segundo y tercer trimestre del 2010 consigue una cuota en estados unidos del 43,6% llegando en 2011 superar el 50% a nivel mundial, desde entonces hasta la actualidad, el porcentaje de Android respecto a Apple no ha parado de crecer.

Dicho esto, a continuación, se expondrán los principales componentes del sistema operativo Android.

Dentro de la arquitectura de Android se pueden destacar los siguientes componentes:

* Núcleo Linux: el núcleo son los cimientos de la arquitectura del sistema operativo que proporciona la unión entre las aplicaciones y el hardware.

Como dato interesante, Google depende mucho del kernel del Linux para desarrollar muchas de sus herramientas, y en este caso no es la excepción, ya que le deja a Linux servicios básicos como gestión de red, memoria, etc…

* Runtime: posee una serie de librerías de base Java Cada aplicación Android corre su propio proceso con su instancia a la máquina virtual Dalvik. Esta máquina ejecutaba hasta la versión 5.0 archivos en formato. dex, pero a partir de esa versión se utilizar el ART, que compila totalmente al momento de instalación de la aplicación.
* Bibliotecas: Principalmente el lenguaje es C o C++. Entre estas bibliotecas, caben destacara System C, bibliotecas de medios, de gráficos, 3 D o SQLite, entre otras.
* Marco del trabajo de aplicaciones: la arquitectura de Android está diseñada para simplificar la reutilización de componentes. Es decir, cualquier aplicación puede publicar sus capacidades y que otras aplicaciones puedan reutilizarlas dentro de unas reglas de seguridad.
* Aplicaciones: Android cuenta con ciertas aplicaciones base que permiten el uso de las funciones básicas de un dispositivo como son, correo electrónico, mensajes de texto SMS, calendario, mapas, navegador, contactos y otros. Aplicaciones desarrolladas en lenguaje Java.

#### **Postman**

Es una herramienta realmente útil para el testeo de los servicios publicados por un api REST o un conjunto de servicios Soa a través de una interfaz gráfica sencilla. A continuación, se expondrá sus características:

* Crear Peticiones: Te permite crear peticiones de diferentes tipos (Get, Post, etc..) para servicio REST, estas peticiones se quedan almacenadas en el historial, además te permite guardar y clasificar las diferentes peticiones para poder comparar los diferentes resultados.
* Definir Colecciones: este programa es capaz de guardar las diferentes peticiones según su tipo de petición o conjunto de prueba.
* Gestionar la Documentación: genera documentación basada en las APIs, pudiéndose hacer publicas
* Entorno colaborativo: permite compartir las APIs mediante el uso de una herramienta colaborativa en Cloud.
* Genera código de invocación: genera código para diferentes lenguajes.
* Establece variable: permite establecer tanto una como varas variables dentro de una petición. Estas pueden ser locales o globales.
* Crea Mokups: herramientas para testear las partes desarrolladas.
* Soporta Ciclo Vida API management, Se puede gestionar el ciclo de vida del API Management, desde la conceptualización del API, la definición del API, el desarrollo del API y la monitorización y mantenimiento del API

#### **Modelo vista controlador**

Es un estilo de arquitectura de software que realiza una división en el código, dejando bien claras las diferentes partes implicadas en la aplicación.

Se trata de un modelo muy maduro que ha demostrado su validez, siendo empleado en múltiples aplicaciones, sobre diversos lenguajes, en infinidad de plataformas de desarrollo.

Las partes de la arquitectura serian:

* El Modelo: El modelo representa los datos, la lógica del sistema y los mecanismos de persistencia.

Es responsable de:

* + Acceder al sistema encargado de almacenar los datos (BBDD). En el mejor de los casos, el modelo debe ser independiente del sistema de almacenamiento
  + Define las reglas del negocio, es decir, en el estarán estructuradas las funciones que ejecutara el sistema para que el programa ofrezca los resultados deseables.
  + Lleva un registro de las vistas y controladores del sistema
  + Si es un modelo activo, debe notificar los cambios producidos en los sistemas de almacenamiento por un agente externo, a las vistas asociadas
* Las Vistas: es la interfaz de usuario, que se encarga de mostrar la información al cliente y de capturar la interacción sobre esta

Es responsable de:

* + Recibir la información del modelo y mostrarla al usuario
  + Captura la respuesta del usuario ante la información mostrada y remite dicha respuesta al modelo
  + Tiene un registro del controlador asociado
* El Controlador: es el encargado de hacer de intermediario entre la vista y el modelo, controlando el flujo de mensajes y adaptándolos a las necesidades de cada uno.

Es responsable de:

* + Recibe la información de la vista
  + Contiene reglas de gestión, las cuales siempre llamaran a funciones de las vistas o del modelo.

El flujo dentro de esta arquitectura seria:



### Back-end

Como ya se presento en el apartado de tecnología, esta parte es la encargada de soportar el grueso de la aplicación, por lo general esta parte se encuentra alojada en un servidor y no genera ninguna interfaz gráfica.

#### **Java**

En 1991 Sun Microsystems creó el lenguaje de programación “Oak” (lenguaje de programación para dispositivos digitales de consumo. Posteriormente se llamó “Green”, como el equipo que lo desarrollo dentro de esa empresa, y al final acabó llamándose “Java” del café.

Para hacer frente al lenguaje predominante en el sector (C/C++) creo este lenguaje con un objetivo claro “escribe una vez, ejecútalo en cualquier lado”.

Más tarde en 2010 Oracle adquirió Sun Microsystems junto con Java.

Una vez que ya se conoce un poco la historia, a continuación, se mostraran sus características:

* Java es independiente a la plataforma: Java fue creado con la filosofía de que se pueda ejecutar en cualquier lado sin modificaciones, para ello se sirve de una maquina virtual que es la encargada de traducir ese lenguaje a el lenguaje maquina de la maquina en la que este instalada.
* Un lenguaje orientado a objetos: es uno de los estilos de programación mas popular, esto se debe a que, en la programación, un programa complejo se subdivide en problemas más simples, los cuales serán tratados mediante objetos. Esto hace que el código sea reutilizable y sea más fácil de mantener.
* Java es rápido: En versiones antiguas, Java era bastante lento, pero este ya no es un problema con las ultimas actualizaciones de las máquinas virtuales, siendo casi tan rápido como los lenguajes C y C++, siendo más rápido que Python, PHP, etc…
* Java es seguro: Proporciona una plataforma segura para desarrollar y ejecutar aplicaciones, además proporciona que la comunicación sea segura al proteger la integridad y privacidad de los datos compartidos, por último, administra de forma de forma automática la memoria, reduciendo las vulnerabilidades y corrupción de esta.

Además, Java ofrece la posibilidad de desarrollar en diferentes aplicaciones como Android, web, Big Data, etc.

#### **Raspbian**

Es una distribución de Linux basado en Debian Strech para la placa computadora Raspberry Pi, orientado a la enseñanza informática.

Técnicamente el sistema operativo es un por no Oficial de Debian armhf para el CPU de Raspberry Pi, con soporte optimizado para coma flotante, dando un mejor rendimiento en algunos casos.

Cabe destacar que la distribución usa LXDE como escritorio y Chromiun como navegador, además como cualquier Linux ya tiene el compilador de Python por defecto.

Por último, no hay que olvidar el menú “raspi-config” que permite configurar el sistema operativo sin tener que modificar los ficheros de configuración manualmente.

### Bases de datos

Según Oracle “Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Normalmente, una base de datos está controlada por un sistema de gestión de bases de datos (DBMS). En conjunto, los datos y el DBMS, junto con las aplicaciones asociadas a ellos, reciben el nombre de sistema de bases de datos, abreviado normalmente a simplemente base de datos.”[[4]](#footnote-4)

Cabe destacar que también depende de los datos que almacene, la estructura de la base de datos puede ser de diferente tipo:

* Según la variabilidad de la base de datos
  + Estáticas: almacenar y registrar datos históricos para poder analizarlos a lo largo del tiempo
  + Dinámicas: actualización, edición y eliminación de los datos utilizadas para un seguimiento de entes vivos en los registros, como por ejemplo la cuenta bancaria de un cliente
* Según el contenido
  + Bases de Datos Bibliográficas: Actúan como un registro cuya función principal sería el clasificar diversos campos de datos.
  + Bases de datos de texto completo: permite buscar un texto completo almacenado, gracias a que este se encuentra relacionado con una serie de palabras claves.
  + Directorios: bases de datos de uso cotidiano que poseen generalmente una copia de seguridad en la nube.
* Según los modelos de bases de datos
  + Bases de datos jerárquicas: estructuradas en un diagrama de árbol
  + Base de datos de red: similar a las jerárquicas, pero en este caso el nodo puede tener diversos padres
  + Bases de datos Transaccionales: Se encargan del envío y recepción de datos a gran velocidad. Son poco comunes
  + Bases de datos relacionales: se almacenan en formato de tablas relacionadas mediante claves.
  + Bases de datos multidimensionales: posen los datos almacenados en diferentes dimensiones o métricas
  + Base de datos orientadas a objetos: almacena el objeto como tal, no solo la información de este.
  + Bases de datos documentales: permiten el manejo de grandes volúmenes de información en periodos mínimos de tiempo.
  + Bases de datos deductivas: se sirve de reglas lógicas para optimizar las diversas consultas.

En este caso las bases de datos que mas se ajustaban a las necesidades del proyecto eran MySQL y MariaDB, las cuales analizaremos en los siguientes apartados.

#### **MySQL**

Es un sistema de bases de datos relacionales con un modelo cliente-servidor que es capaz de almacenar una determinada capacidad de datos de forma permanente.

Para poder entenderlo mejor, a continuación, se expondrán sus características:

* Arquitectura Cliente y Servidor: Esto hace referencia a que tanto como el cliente y el servidor quedan diferenciados, donde el cliente puede realizar consultas al servidor pidiéndole una serie de datos y enviarle una respuesta.
* Compatibilidad con SQL: Al ser este un lenguaje dentro de la industria, este posee plena compatibilidad con el mismo.
* Vistas: es una visión de una consulta que es almacenada en cache con el objetivo de agilizar las búsquedas, esto es muy útil cuando la base de datos almacena una gran cantidad de datos.
* Procedimientos almacenados: MySQL pose la capacidad de no procesar las tablas como tal, ya que seria muy ineficiente, sino, que realiza una serie de pasos con el fin de conseguir esa eficiencia.
* Desencadenantes: Posee la capacidad de crear tareas que reaccionan a determinados eventos producidos dentro de la misma base de datos.
* Transacciones: Representa un conjunto de acciones o interactuaciones de un cliente o usuario respecto a una base de datos.

#### **MariaDB**

Es un sistema de gestión de bases de datos. Se deriva de MySQL y las bases de esta base de datos son similares, pero MariaDB tiene una serie de características que hacen sombra a MySQL. Algunas de estas son:

* Mejoras en la velocidad:
  + La tabla de chequeo de redundancia es más rápida
  + Se eliminaron/mejoraron algunas conversiones no necesarias de juegos de caracteres
  + El uso del motor aria permite realizar consultas complejas rápidamente.
* Extensiones y nuevas Características:
  + Puede manejar hasta 32 segmentos clave por clave (sobre los 16 originales)
  + Pool de Hilos de Ejecución o Procesos
  + Extensiones de prueba mysqltest
  + Cache de Claves segmentadas
  + Especificación de motor de almacenamiento en CREATE TABLE
  + Mejoras a la tabla INFORMATION SCHEMA.PLUGINS
  + Precisión de Microsegundos en la lista de Procesos
  + Estadísticas extendidas para el usuario
* Mejores Pruebas:
  + Mas cantidad de pruebas disponibles en la suite de pruebas
  + Errores en las pruebas corregidos
  + Pruebas invalidas removidas
  + Pruebas construidas con diferentes opciones de configuración para obtener mejores resultados
* Mas motores de almacenamiento:
  + Aria
  + XtraDB (reemplazo para InnoDB)
  + PBXT
  + FederatedX (reemplazo para Federated)
  + OQGRAPH
  + SphinxSE
  + TokuDB

## Resumen

La aplicación posee un hueco en el mercado el cual puede aprovechar, pero debe estar al tanto de las tecnologías emergentes ya que es un mercado muy cambiante.

Para poder adaptarse a dichos cambios va a utilizar tecnología web, ya que esta es la única manera de hacerla portable sin abandonar la idea de hacerla fácil de modificar y mantener, para ello el proyecto toma la tecnología REST en vez de SOA, mas que nada por la facilidad de implementación y por los recursos que ofrece manteniendo la simplicidad, cabe destacar que esta es la tecnología mas puntera en cuanto a los servicios web, siendo esto otro punto favorable para REST.

Respecto a la distribución de la aplicación, se seguirán las pautas de la filosofía REST para ello lo va a dividir en torno al Front-end y Back-end, siendo el primero el que se encargue de la parte de el cliente y el segundo de la parte del servidor, haciéndolos mas estancos y más fáciles de actualizar o modificar.

Dentro del Front-end. Se opto por Android ya que el dispositivo que mas se usa actualmente es el móvil y dentro de los móviles el mas usado es el que tiene el sistema operativo Android.

Respecto a Postman es una tecnología que va a ser utilizada para el testeo de los servicios publicados, pudiendo ver así si su funcionamiento es correcto.

Por parte del Back-end se van a desarrollar los servicios en Java dado que es el lenguaje en el que puedes desarrollar la parte del Front del cliente en Android, además de ser el lenguaje en el cual poseo más experiencia es un lenguaje orientado a objetos que es totalmente compatible con la filosofía REST.

Si nos metemos a la sección de Raspbian, su empleo es simple, ya que es necesario una maquina que actúe como servidor y en este caso opte por la Raspberry Pi ya que es un ordenador barato, fácil de manipular y en él se puede levantar servicios como apache Tomcat y MariaDB que son claves para el correcto funcionamiento de la aplicación.

Por último, opte por estas dos bases de datos por que son SQL, al ser de ese tipo me brindan la estructura que necesita el proyecto.

Además, al Ser MariaDB una BBDD derivada de MySQL facilita muchísimo la compatibilidad entre ambas pudiendo disponer de MySQL para el entorno de pruebas en local y MariaDB en el servidor.

# Análisis

El análisis contendrá, al menos, la información que se muestra a continuación:

* Todas aquellas secciones que se consideren oportunas de estado de la cuestión, lagunas a cubrir en el estado de la cuestión, etc.
* Especificación de requisitos, que incluirá los siguientes puntos (los que proceda):
  + Requisitos funcionales.
  + Requisitos de interfaces externas. Las que proceda de las siguientes: interfaz de usuario, interfaces software e interfaces de comunicación.
  + Requisitos de rendimiento.
  + Requisitos tecnológicos: requisitos de hardware, sistema operativo, virtualización, etc.
* Diagrama de contexto de la aplicación donde se indiquen las entradas y las salidas.
* Diagrama de bloques funcionales con sus interacciones. Tanto los bloques como las interacciones deben estar descritos en el texto de la sección. Los bloques que requieran más detalle pueden, a su vez, mostrarse en diagramas de bloques funcionales de nivel inferior. Estos diagramas no contendrán detalles sobre la tecnología a utilizar (tales detalles se dejan para el diseño).

## Apartado 3.1

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

## Apartado 3.2

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

# Diseño e implementación

Se incluirá un diagrama con la arquitectura del sistema, mostrando cada uno de los componentes y sus interacciones. Tanto los componentes como sus interacciones estarán descritos en el texto del capítulo.

Para aquellos componentes que lo requieran, se incluirán figuras con más detalle y sus explicaciones correspondientes.

Deberán justificarse las decisiones tomadas sobre la tecnología utilizada y su configuración.

El código estará disponible en un repositorio al que tendrán acceso, al menos las siguientes personas:

* El director del proyecto.
* Cada uno de los miembros del tribunal.
* El profesor coordinador de la asignatura de TFG.

El código deberá estar adecudamente documentado. El respositorio deberá incluir los ficheros README.md necesarios con explicaciones que permitan que cualquier desarrollador pueda entender el código, ejecutarlo, mantenerlo y crear nuevas versiones.

De hecho, debe ser posible para los miembros del tribunal clonar el repositorio y probar el sistema. Para ello, el alumno deberá utilizar los recursos de virtualización que sean necesarios.

En la memoria debe aparecer el enlace al repositorio. En caso de que se considere oportuno, se puede separar la implementación en capítulo aparte.

## Apartado 4.1

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

## Apartado 4.2

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

# Evaluación

Se incluirá cómo se han realizado las pruebas del sistema y, en general, cómo se ha evaluado.

## Apartado 5.1

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

## Apartado 5.2

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

## Apartado 5.2

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

# Conclusiones y líneas futuras

Ideas a las que se llega después del desarrollo del proyecto, así como las líneas posibles de trabajo posterior.

## Apartado 6.1

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

## Apartado 6.2

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

# Conclusiones y líneas futuras

Ideas a las que se llega después del desarrollo del proyecto, así como las líneas posibles de trabajo posterior.

## Apartado 6.1

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

## Apartado 6.2

Texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto texto.

# Bibliografía

Bibliografía citada en la memoria. Seguirá el [formato APA](https://apastyle.apa.org/).

Ejemplos de referencias:

1. Jose Manuel Hernández López, Carmen Pozo Muñoz. El fracaso académico en la Universidad: Diseño de un Sistema de Evaluación y detección temprana. Revistas científico-profesionales de psicología del Colegio Oficial de la Psicología de Madrid. Psicología Educativa, 1999, Vol. 5, n.º 1 - Pág. 27-40. https://journals.copmadrid.org/psed/archivos/47547.pdf#page=1&zoom=auto,-20,801
2. Schaefer, N. K., & Shapiro, B. (2019, September 6). New middle chapter in the story of human evolution. Science, 365(6457), 981–982. <https://doi.org/10.1126/science.aay3550>
3. Carey, B. (2019, March 22). Can we get better at forgetting? The New York Times. <https://www.nytimes.com/2019/03/22/health/memory-forgetting-psychology.html>
4. Rabinowitz, F. E. (2019). Deepening group psychotherapy with men: Stories and insights for the journey. American Psychological Association. <https://doi.org/10.1037/0000132-000>
5. Aron, L., Botella, M., & Lubart, T. (2019). Culinary arts: Talent and their development. In R. F. Subotnik, P. Olszewski-Kubilius, & F. C. Worrell (Eds.), The psychology of high performance: Developing human potential into domain-specific talent (pp. 345–359). American Psychological Association. <https://doi.org/10.1037/0000120-016>
6. Harvard University. (2019, August 28). Soft robotic gripper for jellyfish [Video]. YouTube. <https://www.youtube.com/watch?v=guRoWTYfxMs>
7. APA Databases [@APA\_Databases]. (2019, September 5). Help students avoid plagiarismWeb emoji of crossing hands and researchers navigate the publication process. More details available in the 7th edition @APA\_Style table [Tweet]. Twitter. <https://twitter.com/APA_Databases/status/1169644365452578823>
8. Fagan, J. (2019, March 25). Nursing clinical brain. OER Commons. Retrieved September 17, 2019, from <https://www.oercommons.org/authoring/53029-nursing-clinical-brain/view>
9. National Institute of Mental Health. (2018, July). Anxiety disorders. U.S. Department of Health and Human Services, National Institutes of Health. <https://www.nimh.nih.gov/health/topics/anxiety-disorders/index.shtml>
10. Woodyatt, A. (2019, September 10). Daytime naps once or twice a week may be linked to a healthy heart, researchers say. CNN. <https://www.cnn.com/2019/09/10/health/nap-heart-health-wellness-intl-scli/index.html>
11. World Health Organization. (2018, May 24). The top 10 causes of death. <https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death>
12. Datos de estudiante internacionales <https://stats.oecd.org/Index.aspx?DataSetCode=RFOREIGN>

# Glosario de términos

Si es necesario.

# Ilustraciones

Gráfico, Gráfico de barras, Histograma

Descripción generada automáticamente

Gráfico.1 IIntroducción\_estudios\_extranjero\_OCDE. Elaboración propia. Fuente: poner url

Tabla

Descripción generada automáticamente

Tabla 2. Diferencia de medias entre el grupo de fracaso y el grupo de éxito (pag X)

# Anexos

Si son necesarios.

1. <https://www.europapress.es/economia/noticia-inversion-cloud-triplicara-espana-2023-cuando-alcanzara-2800-millones-idc-20191219133342.html> [↑](#footnote-ref-1)
2. <https://www.pcworld.es/articulos/smartphones/iphone-vs-android-cuota-de-mercado-3692825/> [↑](#footnote-ref-2)
3. Un grupo de 84 empresas de tecnología y dispositivos móviles que se han unido para acelerar la innovación en dispositivos móviles y ofrecer a los consumidores una experiencia móvil más rica, menos costosa y mejor. [↑](#footnote-ref-3)
4. https://www.oracle.com/es/database/what-is-database.html [↑](#footnote-ref-4)