00 - Introducción a Desarrollo Web

Ing. Julián Quimbayo

Ingeniería de Sistemas

Facultad de Ingeniería

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"

CICLO DE VIDA DE SOFTWARE

SOFTWARE LIFE CYCLE

El ciclo de vida del desarrollo del software (también conocido como SDLC o Systems Development Life Cycle) contempla las fases necesarias para validar el desarrollo del software y así garantizar que este cumpla los requisitos para la aplicación y verificación de los procedimientos de desarrollo, asegurándose de que los métodos usados son apropiados.

Fuente: https://fundamentossoftware2.blogspot.com/2015/03/modelado-del-software-y-su-importancias.html

CICLO DE VIDA DE SOFTWARE

SOFTWARE LIFE CYCLE

Aunque existen diferentes ciclos de desarrollo de software, la normativa ISO/IEC/IEEE 12207:2017 establece:

“Un marco común para los procesos del ciclo de vida de los programas informáticos, con una terminología bien definida, a la que pueda remitirse la industria del software. Contiene procesos, actividades y tareas aplicables durante la adquisición, el suministro, el desarrollo, el funcionamiento, el mantenimiento o la eliminación de sistemas, productos y servicios informáticos. Estos procesos del ciclo de vida se llevan a cabo mediante la participación de los interesados, con el objetivo final de lograr la satisfacción del cliente”.

Fuente: https://www.timetoast.com/timelines/metodologias-agiles-y-tradicionales-28914048-ffbf-4cfb-addd-b9150612f18e

Fases de Desarrollo de Software

Software development phases

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con grandes posibilidades de éxito. De esta forma, las etapas del desarrollo de software son las siguientes:

Planificación

Antes de empezar un proyecto de desarrollo de un sistema de información, es necesario hacer ciertas tareas que influirán decisivamente en el éxito del mismo. Dichas tareas son conocidas como el *fuzzy front-end* del proyecto, puesto que no están sujetas a plazos.

Algunas de las tareas de esta fase incluyen actividades como la determinación del ámbito del proyecto, la realización de un estudio de viabilidad, el análisis de los riesgos asociados, la estimación del coste del proyecto, su planificación temporal y la asignación de recursos a las diferentes etapas del proyecto.

Fuente: https://hcdsoluciones.com/tag/accion-comercial/

Fases de desarrollo de software

Software development phases

Análisis

Por supuesto, hay que averiguar qué es exactamente lo que tiene que hacer el software. Por eso, la etapa de análisis en el ciclo de vida del software corresponde al proceso a través del cual se intenta descubrir qué es lo que realmente se necesita y se llega a una comprensión adecuada de los requerimientos del sistema (las características que el sistema debe poseer).

Fuente: pixabay.com

Fases de desarrollo de software

Software development phases

Diseño

En esta fase se estudian posibles opciones de implementación para el software que hay que construir, así como decidir la estructura general del mismo. El diseño es una etapa compleja y su proceso debe realizarse de manera iterativa. Es posible que la solución inicial no sea la más adecuada, por lo que en tal caso hay que refinarla. No obstante, hay catálogos de patrones de diseño muy útiles que recogen errores que otros han cometido para no caer en la misma trampa.

Fuente: pixabay.com

Fases de desarrollo de software

Software development phases

Pruebas

Como errar es humano, la fase de pruebas del ciclo de vida del software busca detectar los fallos cometidos en las etapas anteriores para corregirlos. Por supuesto, lo ideal es hacerlo antes de que el usuario final se los encuentre. Se dice que una prueba es un éxito si se detecta algún error.

Instalación o despliegue

La siguiente fase es poner el software en funcionamiento, por lo que hay que planificar el entorno teniendo en cuenta las dependencias existentes entre los diferentes componentes del mismo.

Es posible que haya componentes que funcionen correctamente por separado, pero que al combinarlos provoquen problemas. Por ello, hay que usar combinaciones conocidas que no causen problemas de compatibilidad.

Fuente: https://www.uchicagomedicine.org/forefront/coronavirus-disease-covid-19/covid-testing-when-to-test

Fases de desarrollo de software

Software development phases

Uso y mantenimiento

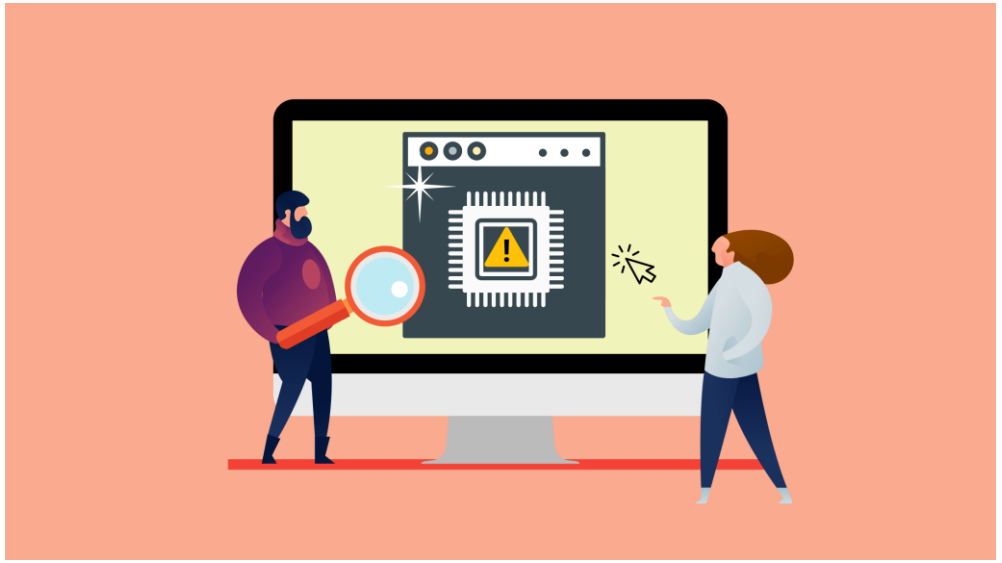
Esta es una de las fases más importantes del ciclo de vida de desarrollo del software. Puesto que el software ni se rompe ni se desgasta con el uso, su mantenimiento incluye tres puntos diferenciados:

•Eliminar los defectos detectados durante su vida útil (mantenimiento correctivo).

•Adaptarlo a nuevas necesidades (mantenimiento adaptativo).

•Añadirle nuevas funcionalidades (mantenimiento perfectivo).

Aunque suene contradictorio, cuanto mejor es el software más tiempo hay que invertir en su mantenimiento. La principal razón es que se usará más (incluso de formas que no se habían previsto) y, por ende, habrá más propuestas de mejoras.



Fuente: pixabay.com

Modelos de ciclo de vida software

Software lifecycle models

Modelo en espiral

El modelo en espiral es una combinación de los modelos anteriores donde se tiene en cuenta el riesgo. De esta forma, se comienza fijando los objetivos y las limitaciones al empezar cada repetición. En la etapa siguiente se crean los modelos de prototipo del software, que incluye el análisis de riesgo. Posteriormente se usa un modelo estándar para construir el software y finalmente se prepara el plan de la próxima repetición.

Fuente: https://commons.wikimedia.org/wiki/File:Software_Development_Spiral_cz.svg

Modelos de ciclo de vida software

Software lifecycle models

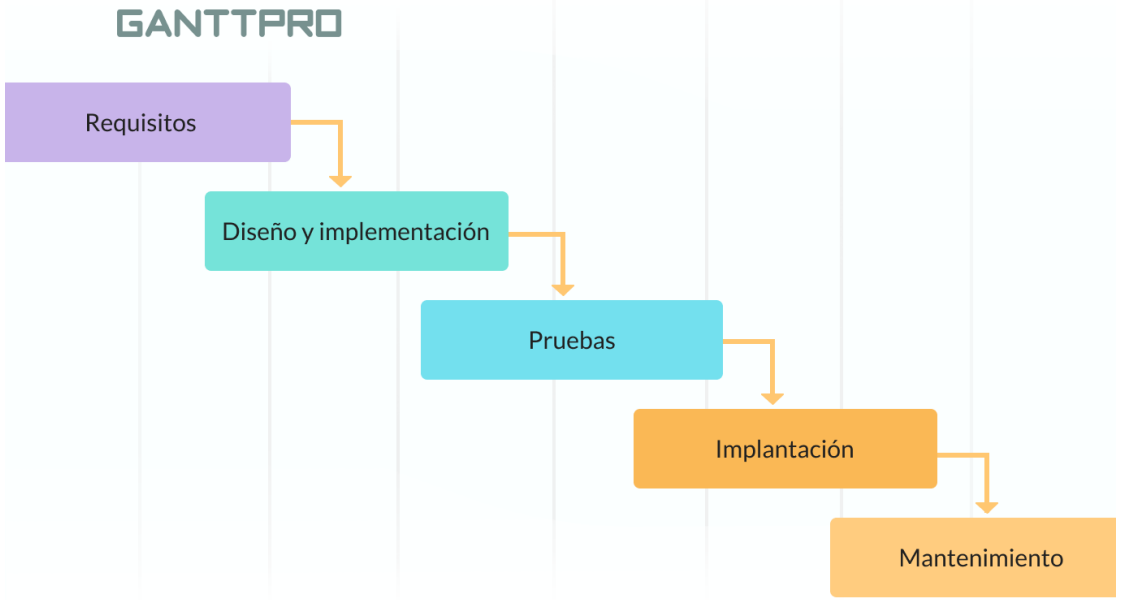
Con el fin de facilitar una metodología común entre el cliente y la compañía de software, los modelos de ciclo de vida (o paradigmas de desarrollo de software como la programación orientada a objetos) se han actualizado para plasmar las etapas de desarrollo involucradas y la documentación necesaria, de forma que cada fase se valide antes de continuar con la siguiente.

Modelo en cascada

En el modelo de ciclo de vida en cascada las fases anteriores funcionarán una detrás de la otra de manera lineal. De este modo, solo cuando una fase termine se podrá continuar con la siguiente, y así progresivamente.

Modelo repetitivo

Este modelo guía el proceso de desarrollo de software en repeticiones. Así, proyecta el proceso de desarrollo de modo cíclico repitiendo cada paso después de cada ciclo en el proceso de ciclo de vida del software.



Fuente: https://blog.ganttpro.com/es/metodologia-de-cascada/

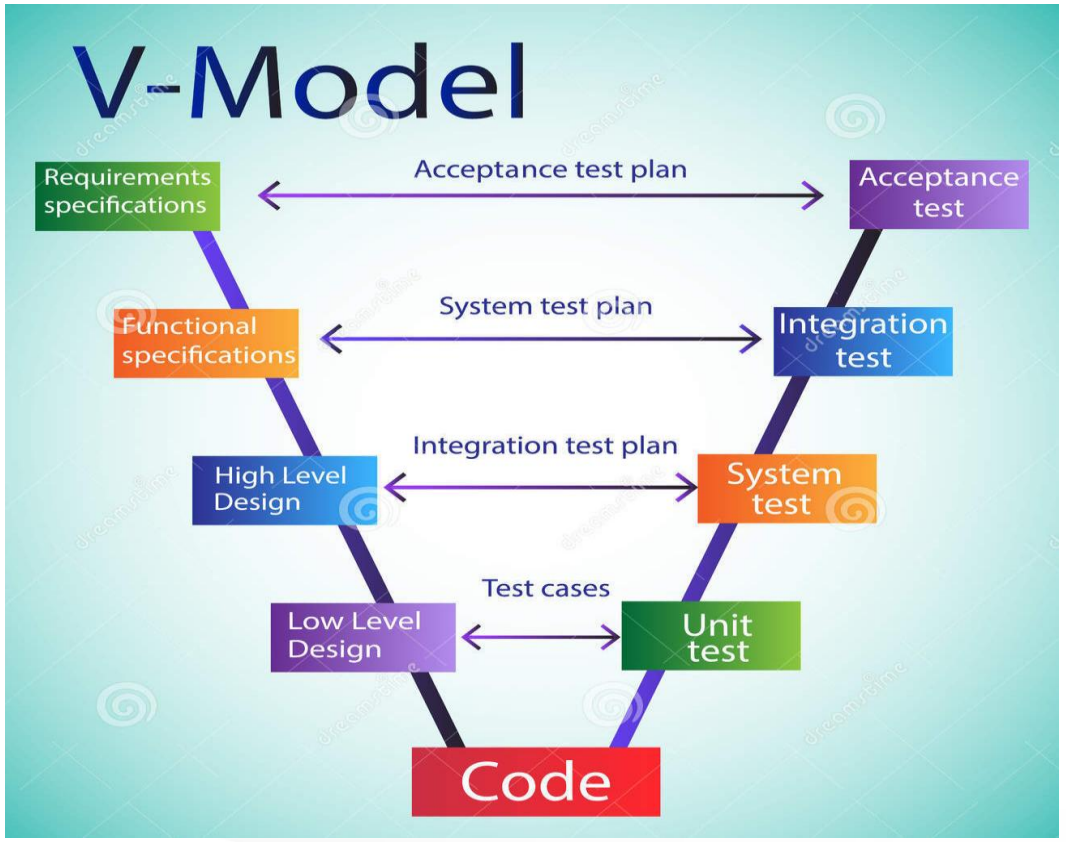
Modelos de ciclo de vida software

Modelo en V

Uno de los grandes problemas del modelo en cascada es que solo se pasa a la siguiente fase si se completa la anterior y no se puede volver atrás si hay errores en etapas posteriores. Así, el modelo en V da más opciones de evaluación del software en cada etapa.

En cada fase se crea la planificación de las pruebas y los casos de pruebas para verificar y validar el producto en función de los requisitos de la misma. De esta manera, verificación y validación van en paralelo.

Software lifecycle models



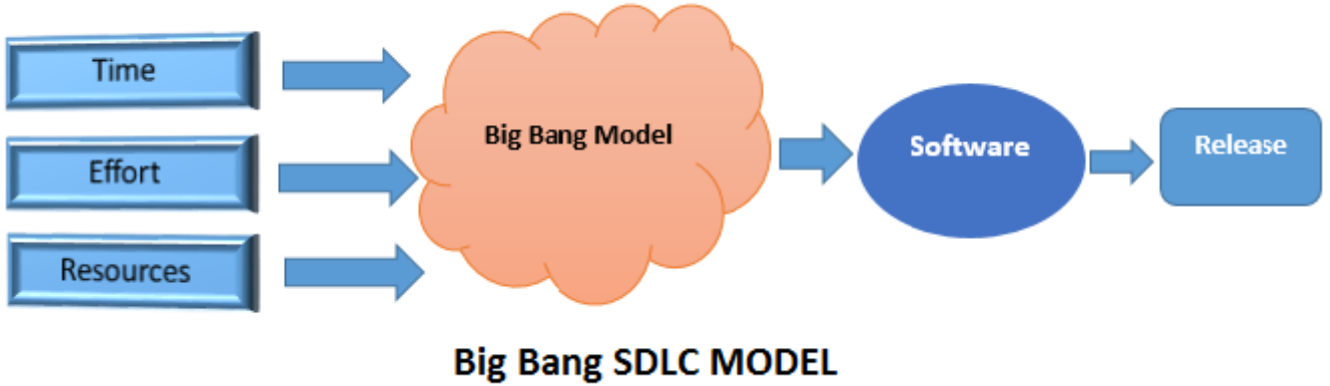
Fuente: https://www.shutterstock.com/es/image-vector/concept-software-development-life-cycle-v-419198509

Modelos de ciclo de vida software

Software lifecycle models

Modelo Big Bang

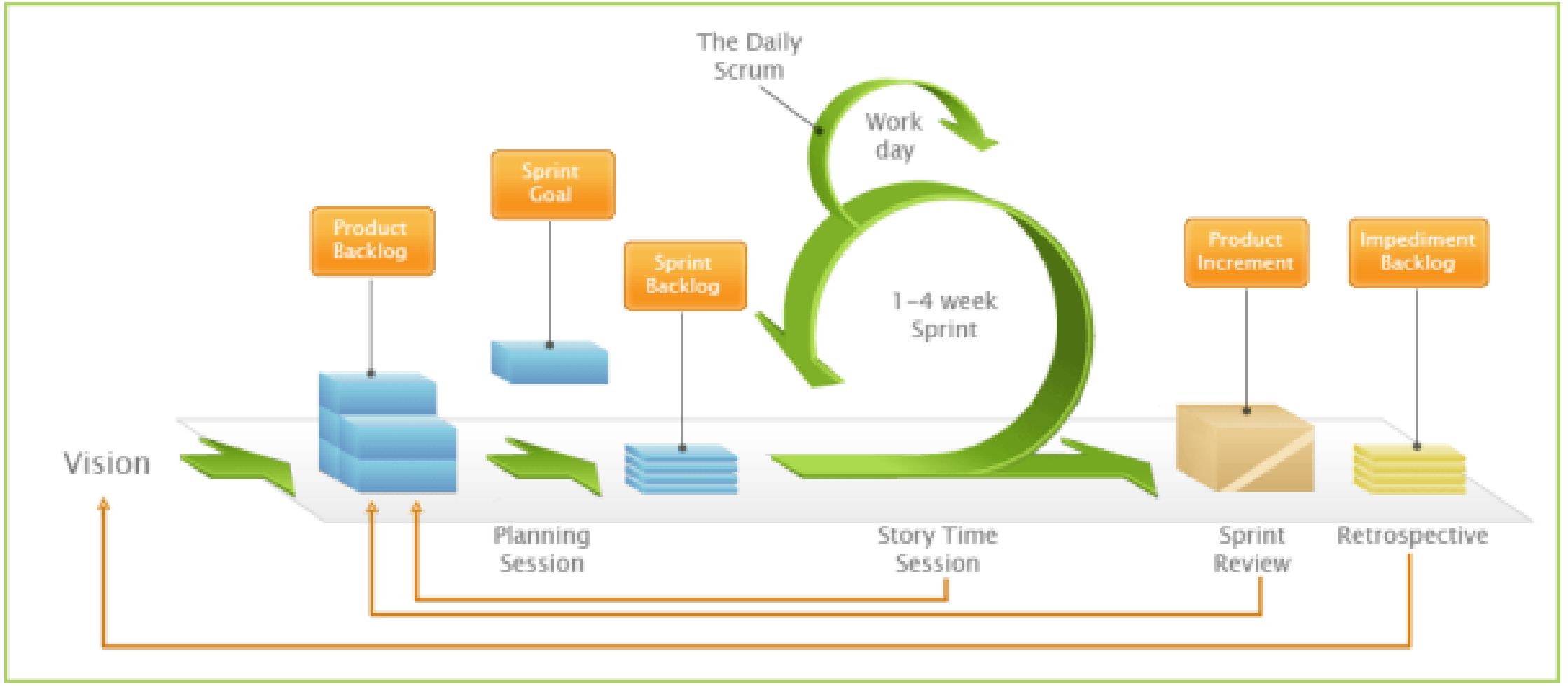
Probablemente este sea el modelo más simple, ya que necesita poca planificación, mucha programación y muchos fondos. Este modelo tiene como concepto principal la creación del universo; así, si se reúnen fondos y programación, se consigue el mejor producto de software.



Big Bang SDLC MODEL

Fuente: https://www.scaler.com/topics/software-engineering/big-bang-model-in-software-engineering/

Scrum



Fuente: http://agilefunding.org/about/about_us/

Modelos de ciclo de vida software

Software lifecycle models

Modelo ágil

El **desarrollo ágil de software** envuelve un enfoque para la toma de decisiones en los proyectos de software, que se refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto. Así el trabajo es realizado mediante la colaboración de equipos autoorganizados y multidisciplinarios, inmersos en un proceso compartido de toma de decisiones a corto plazo.

Fuente: https://apis.ar/

Manifiesto Ágil

Agile Manifest

«Estamos poniendo al descubierto mejores métodos para desarrollar software, haciéndolo y ayudando a otros a que lo hagan. Con este trabajo hemos llegado a valorar:

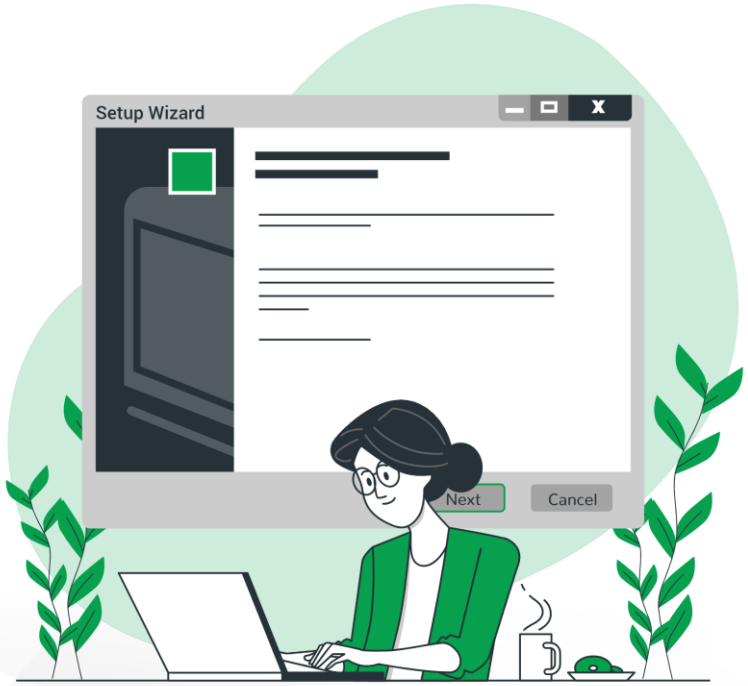
- A los individuos y su interacción, por encima de los procesos y las herramientas.
- El software que funciona, por encima de la documentación exhaustiva.
- La colaboración con el cliente, por encima de la negociación contractual.
- La respuesta al cambio, por encima del seguimiento de un plan.

Fuente: https://donetonic.com/es/que-es-la-gestion-agil-de-proyectos/

12 Principios Ágiles

12 Agile Principles

1. SATISFACER AL CLIENTE – ENTREGA TEMPRANA Y CONTÍNUA



2. REQUISITOS CAMBIANTES – DOBLEGAN AL CAMBIO

Fuente: illustrations by Storyset – Illustraciones por Storyset.com

12 Principios Ágiles

12 Agile Principles

3. PERIODOS BREVES DE ENTREGA

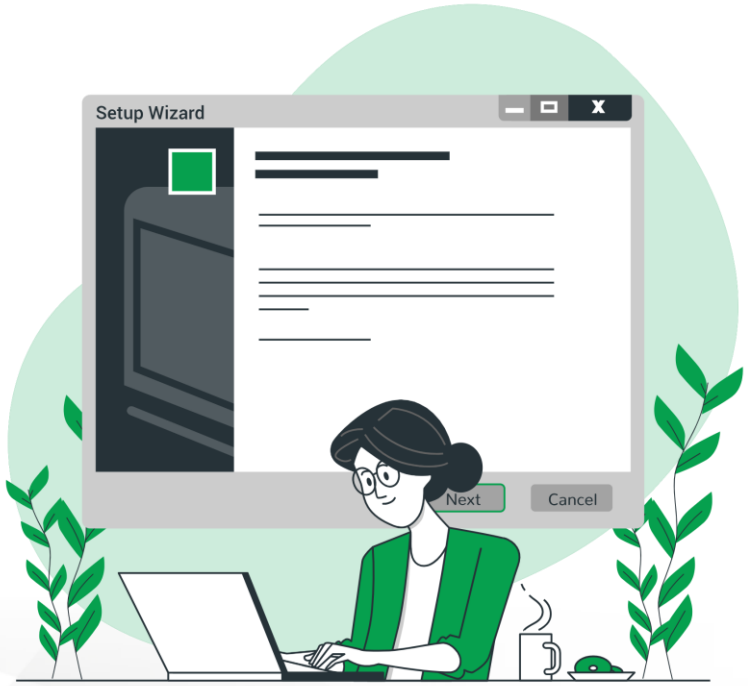
4. TRABAJO CONJUNTO DESARROLLADORES Y NEGOCIO

Fuente: illustrations by Storyset – Illustraciones por Storyset.com

12 Principios Ágiles

12 Agile Principles

5. MOTIVACIÓN Y CONFIANZA AL EQUIPO



6. CONVERSACIÓN CARA A CARA

Fuente: illustrations by Storyset – Illustraciones por Storyset.com

12 Principios Ágiles

12 Agile Principles

7. SOFTWARE QUE FUNCIONA

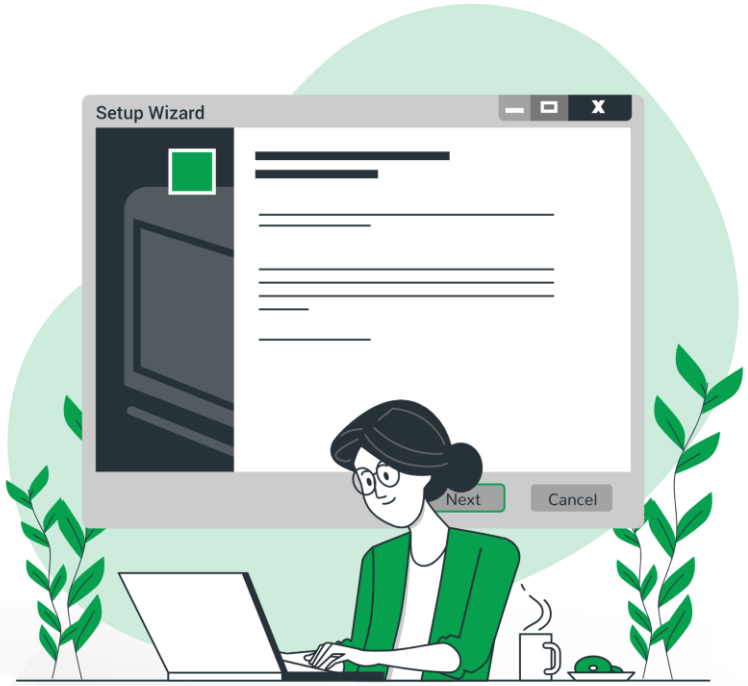
8. RITMO CONSTANTE DEL EQUIPO

Fuente: illustrations by Storyset – Illustraciones por Storyset.com

12 Principios Ágiles

12 Agile Principles

9. ATENCIÓN TÉCNICA = AGILIDAD



10. SIMPLICIDAD = MAXIMIZAR EL TRABAJO

Fuente: illustrations by Storyset – Illustraciones por Storyset.com

12 Principios Ágiles

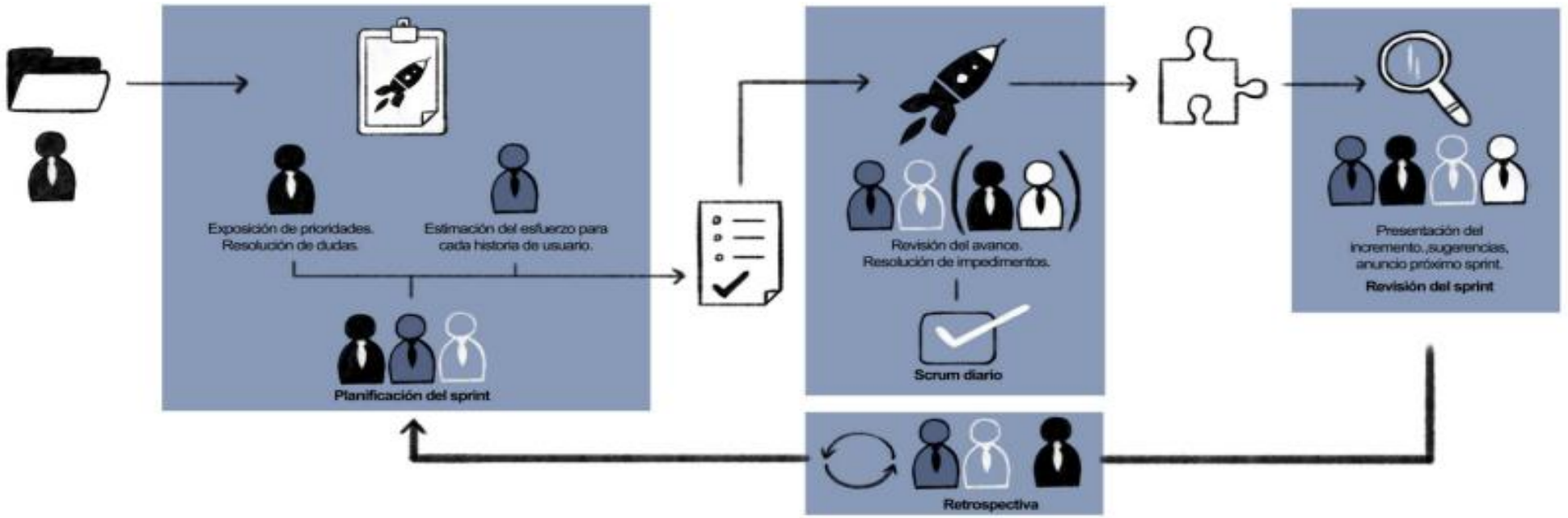
12 Agile Principles

11. EQUIPOS QUE SE AUTOORGANIZAN

12. REFLEXIÓN Y AJUSTE DE CONDUCTA

Fuente: illustrations by Storyset – Illustraciones por Storyset.com

LAS REGLAS DE *SCRUM*



ROLES

PROPIETARIO DEL PRODUCTO
Determina las prioridades.
Una sola persona.

DESARROLLADOR
Construye el producto.

SCRUM MASTER
Gestiona y facilita la ejecución de las reglas del Scrum.

INTERESADOS
Resto de implicados. Asesoran y observan.

ARTEFACTOS

PILA DEL PRODUCTO
Relación de requisitos del producto, no es necesario excesivo detalle. Priorizados. Lista en evolución y abierta a todos los roles. El propietario del producto es su responsable y quien decide.

PILA DEL SPRINT
Requisitos comprometidos por el equipo para el sprint con nivel de detalle suficiente para su ejecución.

INCREMENTO
Parte del producto desarrollada en un sprint, en condiciones de ser usada (pruebas, codificación limpia y documentada).

EVENTOS

PLANIFICACIÓN DEL SPRINT
1 jornada de trabajo (máx.) El propietario del producto explica las prioridades. El equipo estima el esfuerzo de los requisitos prioritarios y se elabora la pila del sprint. El equipo define en una frase el objetivo del sprint.

SPRINT
Ciclo de desarrollo básico en el marco estándar de scrum, de duración recomendada inferior a un mes y nunca mayor de 6 semanas.

SCRUM DIARIO
15 minutos máximo. Responsabilidad del equipo. cada miembro expone: Lo que hizo ayer, lo que va a hacer hoy y si tiene o prevé problemas. Se actualiza la pila del sprint.

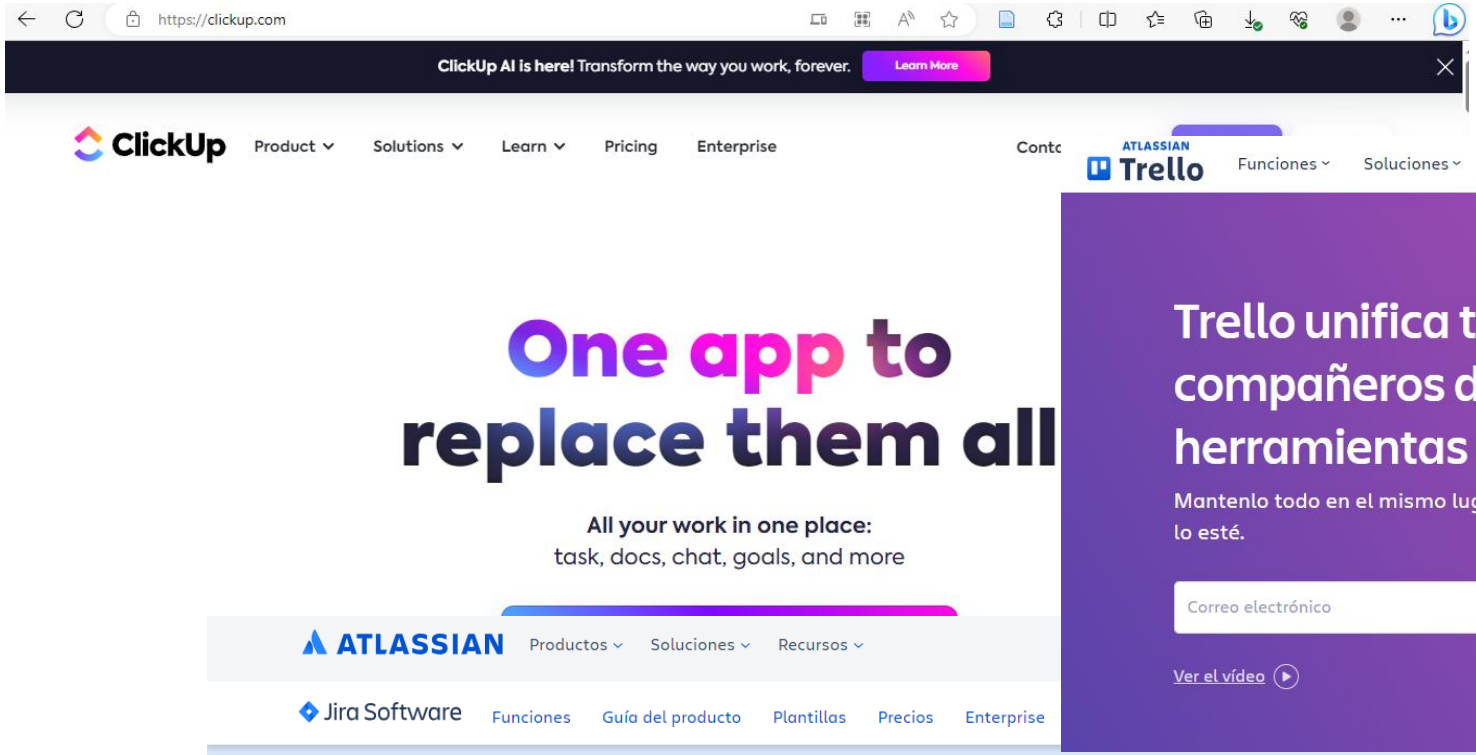
REVISIÓN DEL SPRINT
Informativa, máximo 4 horas. Presentación del incremento, planteamiento de sugerencias y anuncio del próximo sprint.

RETROSPECTIVA
El equipo autoanaliza la forma de trabajo. Identificación de fortalezas y debilidades. Refuerzo de las primeras, plan de mejora de las segundas.

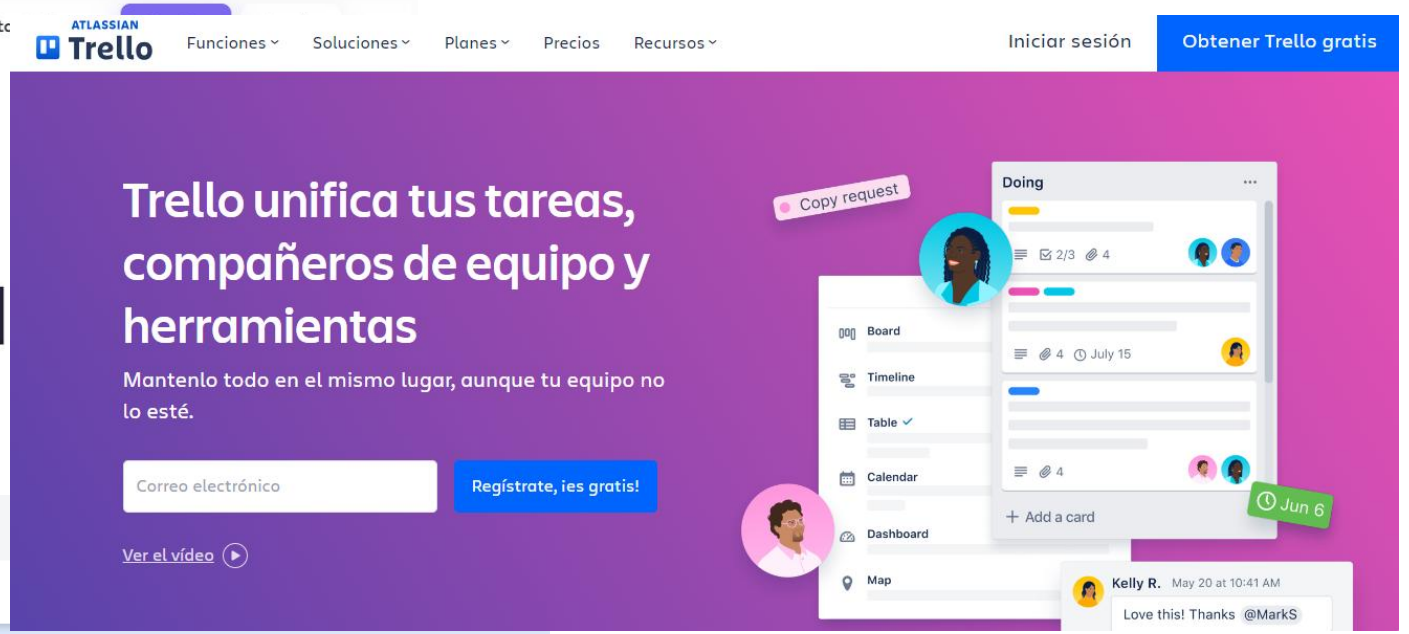
Fuente: https://scrum.menzinsky.com/2018/08/un-poco-de-historia-de-scrum-y-scrum.html

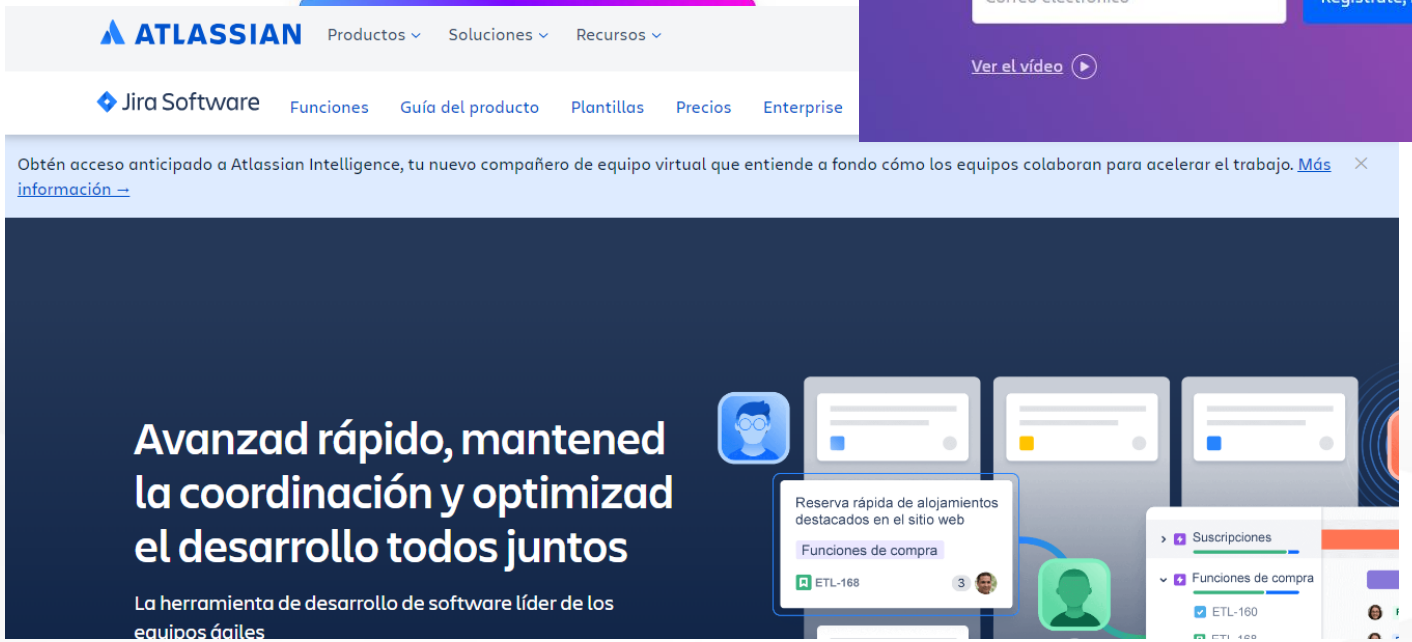
Fuente: https://www.youtube.com/watch?v=HhC75IonpOU

Fuente: https://clickup.com/



Fuente: https://trello.com/es





Fuente: https://www.atlassian.com/es

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA

Especificación de Requerimientos de Software

Software Requirements Specification

Fuente: https://www.youtube.com/watch?v=zB1PuEwojjY

El estándar para la documentación de requerimientos de software se basa en varias prácticas y directrices ampliamente aceptadas que aseguran claridad, consistencia y calidad en la especificación de los requisitos. Un estándar comúnmente utilizado es el IEEE 830-1998, aunque ha sido reemplazado por el ISO/IEC/IEEE 29148:2011, que ofrece una guía integral para la creación de un documento de especificación de requisitos de software (SRS).

Descripción detallada de lo que realmente va a hacer el software. Planificación y análisis.

Especificación de Requerimientos de Software

Software Requirements Specification

El alcance del producto: El alcance debería estar vinculado a los objetivos generales y comerciales del producto, un dato que resultará particularmente importante en caso de que varios equipos o contratistas tengan acceso al documento. Enumera los beneficios, las metas y los objetivos previstos para el producto.

El valor del producto: ¿Por qué tu producto es importante? ¿Qué beneficio le aportará a la audiencia prevista? ¿Qué función tendrá o qué problema resolverá? Pregúntate qué valor encontrará el público en tu producto.

El público objetivo: Describe a tu público ideal. Serán ellos quienes dicten el aspecto que tendrá el producto y cómo deberás comercializarlo.

El uso previsto: Imagina de qué manera tu público usará el producto. Enumera las funciones que brindarás y todas las formas posibles en que la audiencia podrá usar el producto dependiendo del rol. También resultará una buena práctica incluir casos de uso para ilustrar tu visión.

Las definiciones y los acrónimos: Cada sector o negocio tiene sus propios acrónimos y su jerga particular. Dispón las definiciones de los términos que usarás en el documento para asegurarte de que todos entiendan lo que quieres decir.

El índice: Si el documento es exhaustivo, es probable que sea bastante largo. Incluye un índice para ayudar a que los participantes encuentren exactamente lo que buscan.

Especificación de Requerimientos de Software

Software Requirements Specification

Los ***requisitos funcionales*** se desglosan las características y funciones que permiten que el sistema se comporte como ha sido previsto.

Usa el resumen general como referencia para controlar que los requisitos cumplan con las necesidades básicas de los usuarios a medida que completas los datos. Dependiendo del producto, puede haber cientos de requisitos funcionales por incluir. Algunos de los más comunes son los siguientes:

- Funciones “If/then” (si/o)
- Lógica de manejo de datos
- Flujos de trabajo del sistema
- Gestión de las transacciones
- Funciones administrativas
- Deberes regulatorios y de cumplimiento de normas
- Requisitos de desempeño
- Detalles sobre las operaciones llevadas a cabo en cada pantalla

Especificación de Requerimientos de Software

Software Requirements Specification

Los ***requisitos de la interfaz externa*** son tipos de requisitos funcionales con los que se garantiza que el sistema se comunicará correctamente con los componentes externos como:

- **Las interfaces de usuarios:** Son la clave de la facilidad de uso de una aplicación. Incluyen la presentación del contenido, las opciones de navegación de la aplicación y asistencia para los usuarios, entre otros componentes.
- **Las interfaces de hardware:** Son las características de cada interfaz que interactúa entre los componentes del software y del hardware del sistema; como los tipos de dispositivos compatibles y los protocolos de comunicación.
- **Las interfaces de software:** Incluyen las conexiones entre tu producto y otros componentes de software, como las bases de datos, las bibliotecas y los sistemas operativos.
- **Las interfaces para comunicaciones:** Los requerimientos de las funciones que se usarán en tu producto para comunicación, como los emails o los formularios integrados.

Especificación de Requerimientos de Software

Software Requirements Specification

Este ***tipo de requerimientos sirven para controlar los requisitos funcionales***, de modo que se conserven los atributos como la asequibilidad del producto o su facilidad de uso.
Estos son los tipos más comunes de NFR:

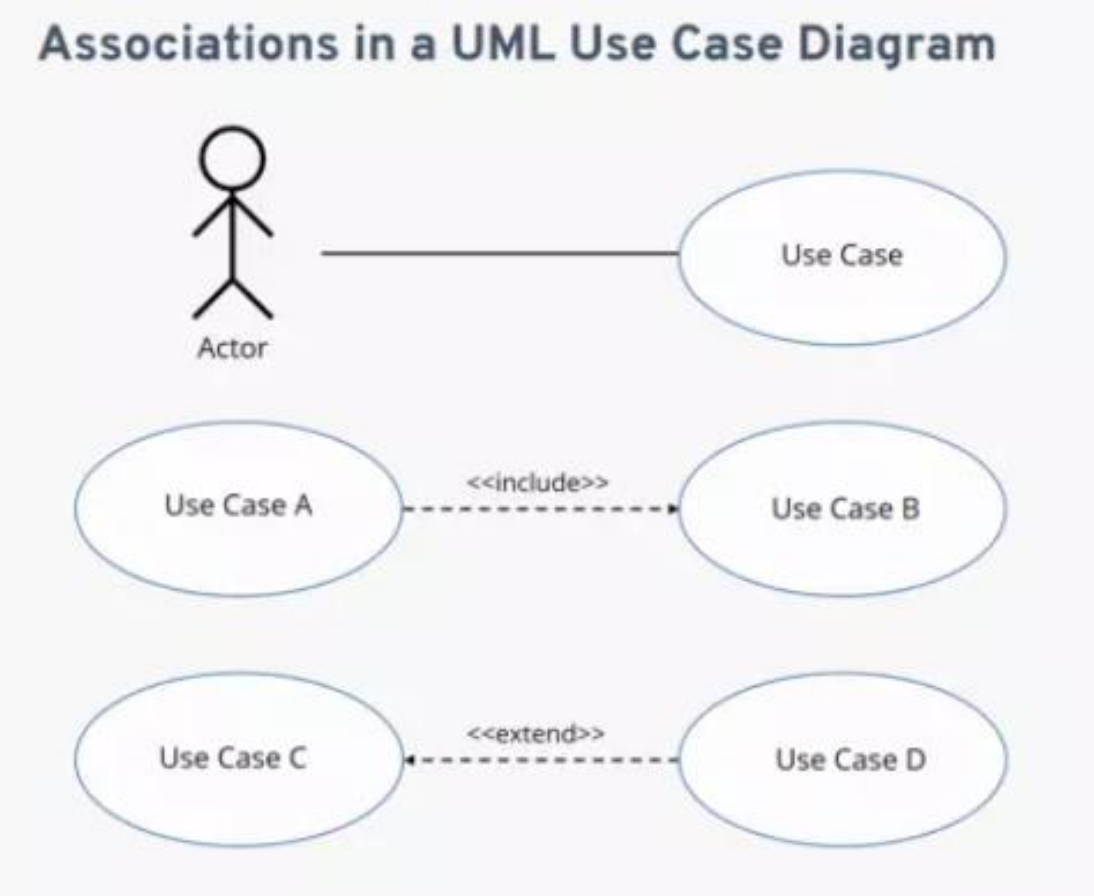
- **La seguridad:** Qué se necesita para garantizar que cualquier información sensible de los usuarios que recopile tu software estará protegida.
- **La capacidad:** Lo que se necesita para satisfacer la demanda de almacenamiento ahora y lo que se necesitará más adelante. Incluye un plan para determinar cómo se adaptará el sistema a escala a medida que aumente la demanda.
- **La compatibilidad:** Los requisitos mínimos de hardware para el software, como la compatibilidad con los sistemas operativos y sus versiones.
- **La fiabilidad y disponibilidad:** Con qué frecuencia esperas que los usuarios utilicen el software y cuál es el tiempo de fallo crítico esperado con relación a un uso normal.
- **La escalabilidad:** La cantidad máxima de trabajo con la que el sistema se comportará como se ha previsto.
- **La mantenibilidad:** Cómo se debería usar la integración continua en tu aplicación para implementar funciones o reparar errores rápidamente.
- **La facilidad de uso:** Cuán fácil resulta usar el producto.

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA

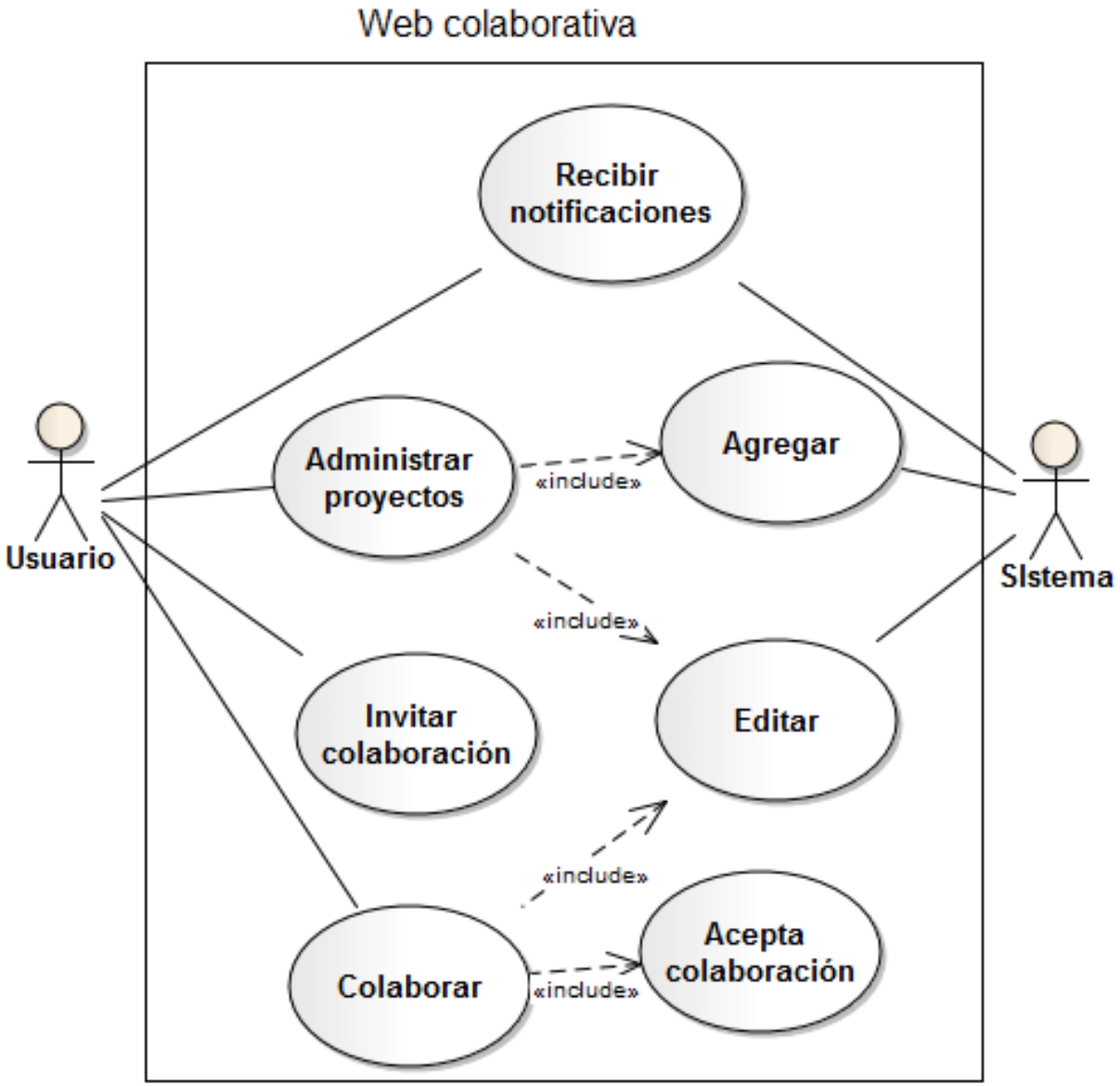
DIAGRAMA DE CASOS DE USO

USE CASE DIAGRAM

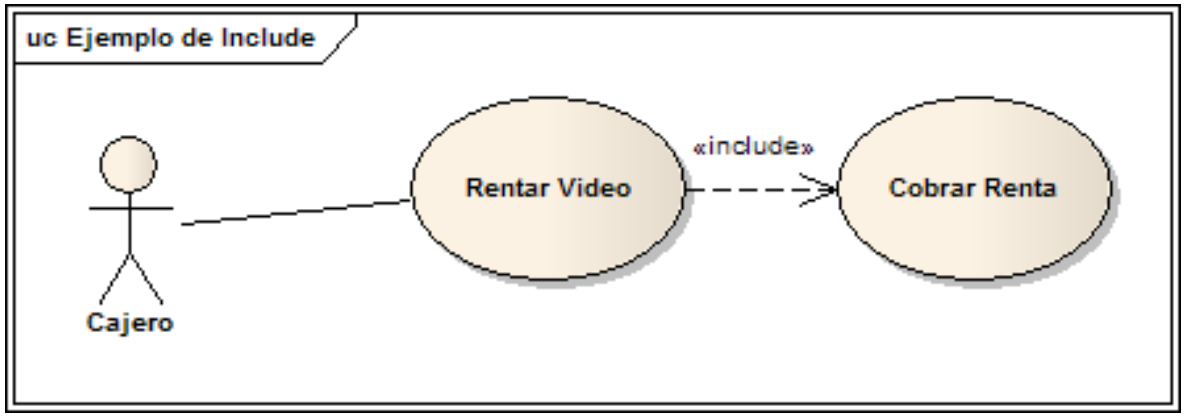


El diagrama de casos de uso es una forma de diagrama de comportamiento en lenguaje de modelado unificado (UML, del inglés Unified Modelling Language), con la que se representan procesos empresariales, así como sistemas y procesos de programación orientada a objetos.

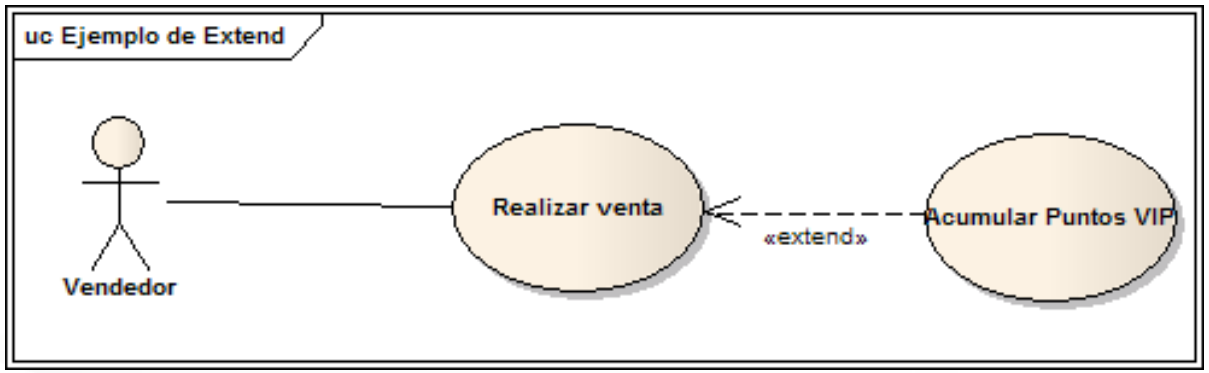
Fuente: https://www.ionos.com/es-us/digitalguide/paginas-web/desarrollo-web/diagrama-de-casos-de-uso/



Fuente: https://www.researchgate.net/profile/Rene-Lopez-5/publication/298787742/figure/fig4/AS:340822261288965@1458269767175/Figura-31-Diagrama-de-casos-de-uso-de-la-Web-colaborativa.png



Fuente: https://www.liderdeproyecto.com/uml/uml016.html

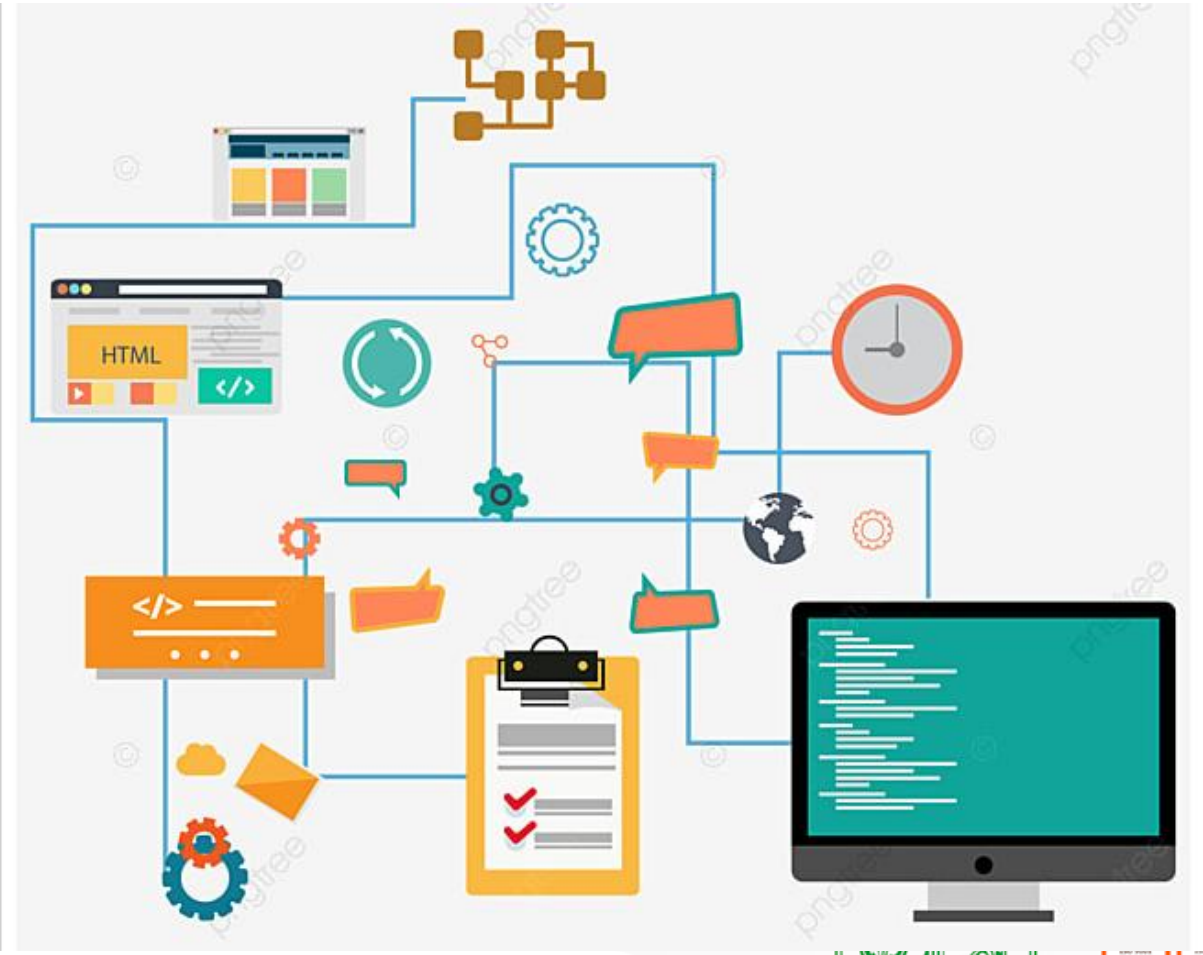


Fuente: https://www.liderdeproyecto.com/uml/uml016.html

¿Qué es la arquitectura de software?

¿What is software architecture?

En palabras simples la arquitectura de software son patrones o lineamientos que ayudan a la construcción de un programa (aplicación). Estos patrones permiten tener una guía para los desarrolladores, analistas y todos los cargos relacionados para lograr cumplir con los requerimientos de la aplicación.



Fuente: https://in.pinterest.com/pin/computer-system-software-vector-art-png--959407526848601517/

Porqué es importante la arquitectura de software

Why software architecture matters

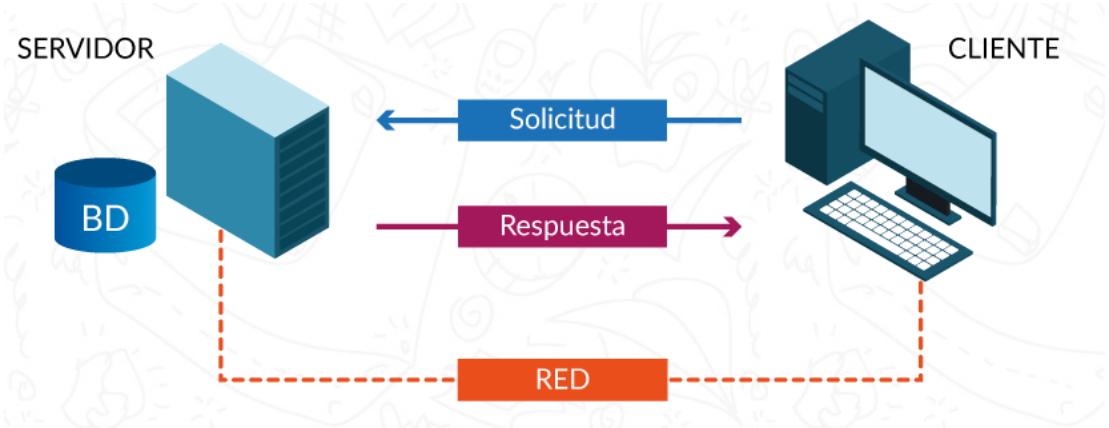
El definir las tecnologías es uno los puntos más importantes de la arquitectura de software pero no quiere decir que si se toma una decisión sea algo definitivo que no se pueda modificar en el futuro. Por ejemplo Uber tenía la parte realtime de los mapas usando node.js, esta fue su primera implementación y funcionaba, llegó un punto donde no estaba escalando de la manera correcta y migraron a usar go para esta parte realtime.
Uno de los objetivos de la arquitectura de software es crear una estructura de la aplicación que sea fácilmente escalable, que no esté fuertemente acoplada (que todo dependa de todo, lo que evita hacer cambios de manera sencilla)

Fuente: https://co.pinterest.com/pin/670754938236404194/

Cliente - Servidor

Client – Server Architecture

Un servidor es una aplicación que ofrece un servicio a usuarios de una red, el servidor puede estar instalada en la misma máquina o puede estar instalada en otra máquina. Es decir un cliente. Es la persona o empresa que nos contrata para desarrollar el software, esto no quiere decir que sea propiamente el que interactúa con el software. Su visión de requerimientos en general es distinta a la de otros agentes, con el cliente podremos obtener requisitos funcionales, requisitos del sistema y del software entre otros. puede ser al mismo tiempo un servidor si el programa esta instalado en la misma máquina. El funcionamiento del modelo cliente/ servidor es sencillo el servidor recibe solicitudes por parte de 1 cliente o distintos clientes, a partir de estas solicitudes, el servidor realiza lo que le solicitan y le devuelve un resultado al cliente.



Fuente: https://contenidos.sucerman.com/nivel4/desarrollo/unidad2/leccion3.html

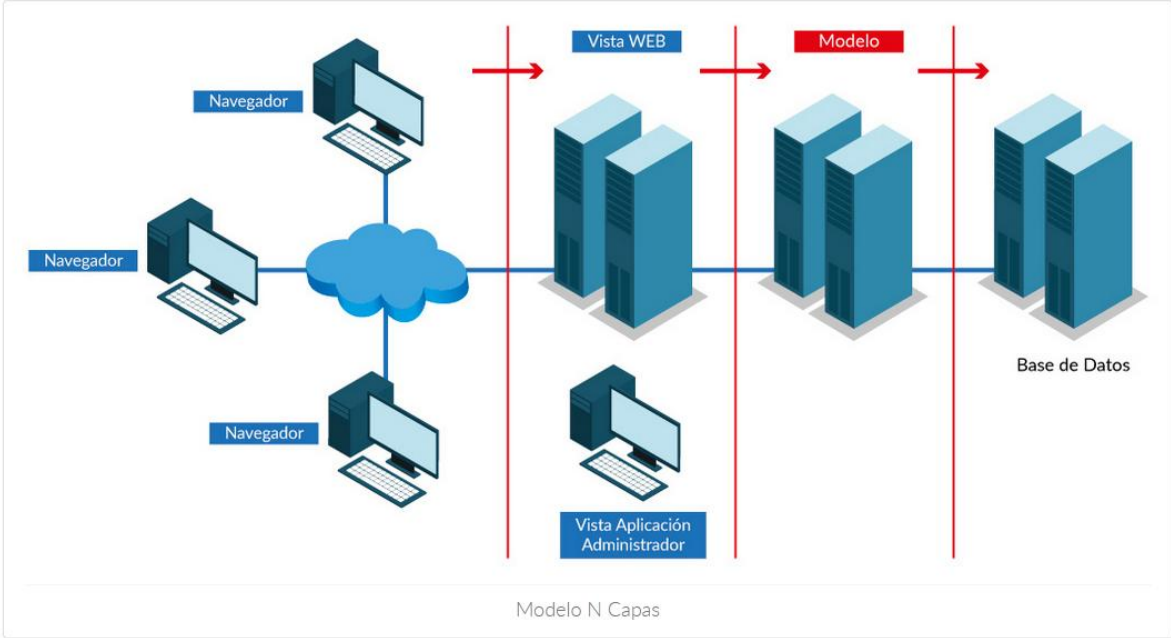
En Capas

Layered

El modelo de capas propone una forma distinta de trabajar el desarrollo web, donde se busca es separar la lógica de negocios de la lógica de diseño, un ejemplo claro puede ser buscar separar la capa de datos de la capa de presentación del usuario. Es la persona o grupos de personas que interactúan directamente con el software y al cual le debemos prestar una atención especial ya que nos brindará principalmente información de requisitos no funcionales, de características inesperadas, requisitos cuantificables entre otros..

Lo que conocemos como capas no es más que un estilo de programación donde utilizamos la máxima “divida y vencerás” y donde el objetivo principal es separar los diferentes aspectos del desarrollo, tales como las cuestiones de presentación, lógica de negocio, mecanismos de almacenamiento, etc.

Esto en resumida permitirá una mejor usabilidad del código del desarrollo web, un mantenimiento más fácil de administrar y un mayor rendimiento del servidor.



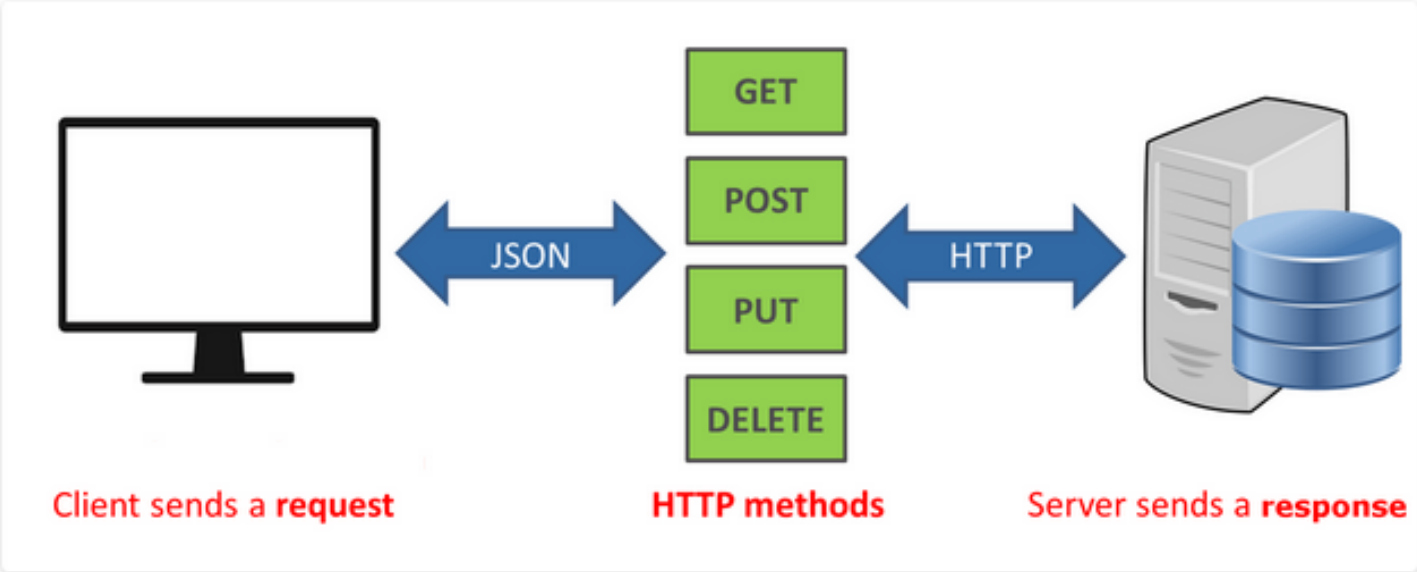
Modelo N Capas

Fuente: https://contenidos.sucerman.com/nivel4/desarrollo/unidad2/leccion3.html

API REST

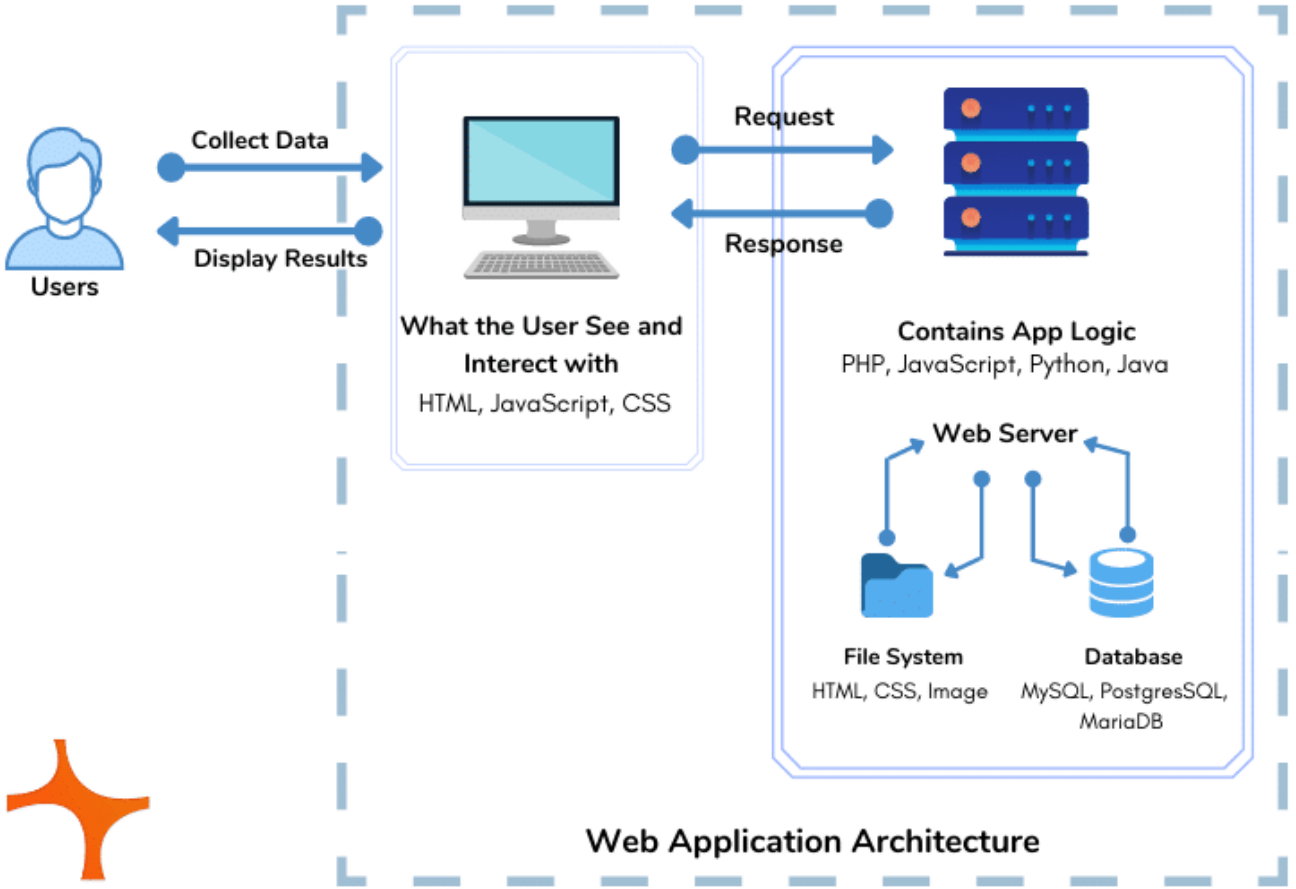
Las API REST se han convertido en una herramienta muy importante para los desarrollos con IoT al permitir conectar servicios entre sí. Vamos a explicar qué son exactamente.

La interfaz de programación de aplicaciones, conocida también por la sigla API, en inglés, application programming interface, es un conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.



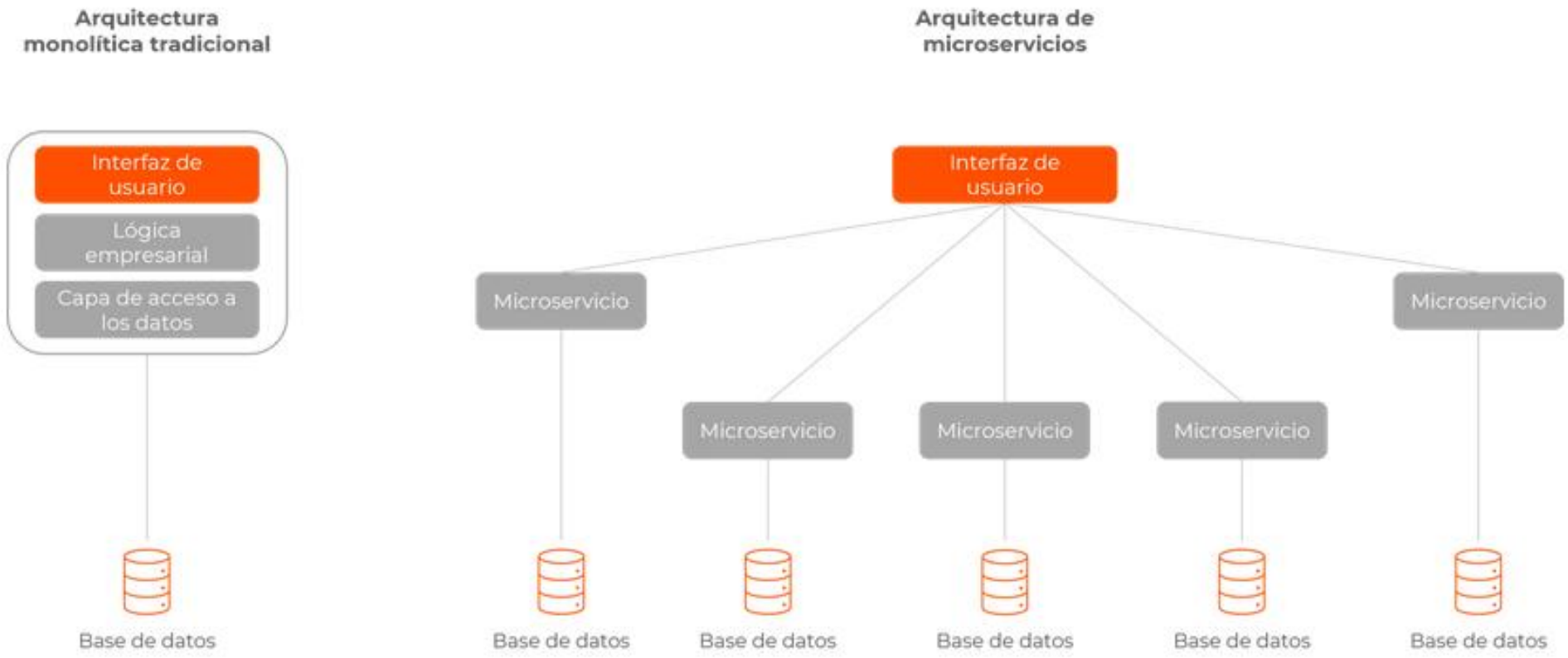
Fuente: https://www.youtube.com/watch?v=l910tH4a2Dc

Web Application Architecture



Fuente: https://achirou.com/pentesting-web-introduccion-a-inyeccion-html/

Arquitectura Micro Servicios



https://consultoria-consultores.es/articulos/articulo-consultoria-nuevos-paradigmas-de-arquitectura-software-en-un-contexto-de-datos-masivos/

Arquitectura Micro Servicios

Micro Service Architecture

•**Modularidad:** al tratarse de servicios autónomos, se pueden desarrollar y desplegar de forma independiente. Además un error en un servicio no debería afectar la capacidad de otros servicios para seguir trabajando según lo previsto.
•**Escalabilidad:** como es una aplicación modular, se puede escalar horizontalmente cada parte según sea necesario, aumentando el escalado de los módulos que tengan un procesamiento más intensivo.
•**Versatilidad:** se pueden usar diferentes tecnologías y lenguajes de programación. Lo que permite adaptar cada funcionalidad a la tecnología más adecuada y rentable.
•**Rapidez de actuación:** el reducido tamaño de los microservicios permite un desarrollo menos costoso, así como el uso de “contenedores de software” permite que el despliegue de la aplicación se pueda llevar a cabo rápidamente.
•**Mantenimiento simple y barato:** al poder hacerse mejoras de un solo módulo y no tener que intervenir en toda la estructura, el mantenimiento es más sencillo y barato que en otras arquitecturas.
•**Agilidad:** se pueden utilizar funcionalidades típicas (autenticación, trazabilidad, etc.) que ya han sido desarrolladas por terceros, no hace falta que el desarrollador las cree de nuevo.

Arquitectura Micro Servicios

Micro Service Architecture

Desventajas

•**Alto consumo de memoria:** al tener cada microservicio sus propios recursos y bases de datos, consumen más memoria y CPU.

•**Inversión de tiempo inicial:** al crear la arquitectura, se necesita más tiempo para poder fragmentar los distintos microservicios e implementar la comunicación entre ellos.

•**Complejidad en la gestión:** si contamos con un gran número de microservicios, será más complicado controlar la gestión e integración de los mismos. Es necesario disponer de una centralización de trazas y herramientas avanzadas de procesamiento de información que permitan tener una visión general de todos los microservicios y orquesten el sistema.

•**Perfil de desarrollador:** los microservicios requieren desarrolladores experimentados con un nivel muy alto de experiencia y un control exhaustivo de las versiones. Además de conocimiento sobre solución de problemas como latencia en la red o balanceo de cargas.

•**No uniformidad:** aunque disponer de un equipo tecnológico diferente para cada uno de los servicios tiene sus ventajas, si no se gestiona correctamente, conducirá a un diseño y arquitectura de aplicación poco uniforme.

•**Dificultad en la realización de pruebas:** debido a que los componentes de la aplicación están distribuidos, las pruebas y test globales son más complicados de realizar.

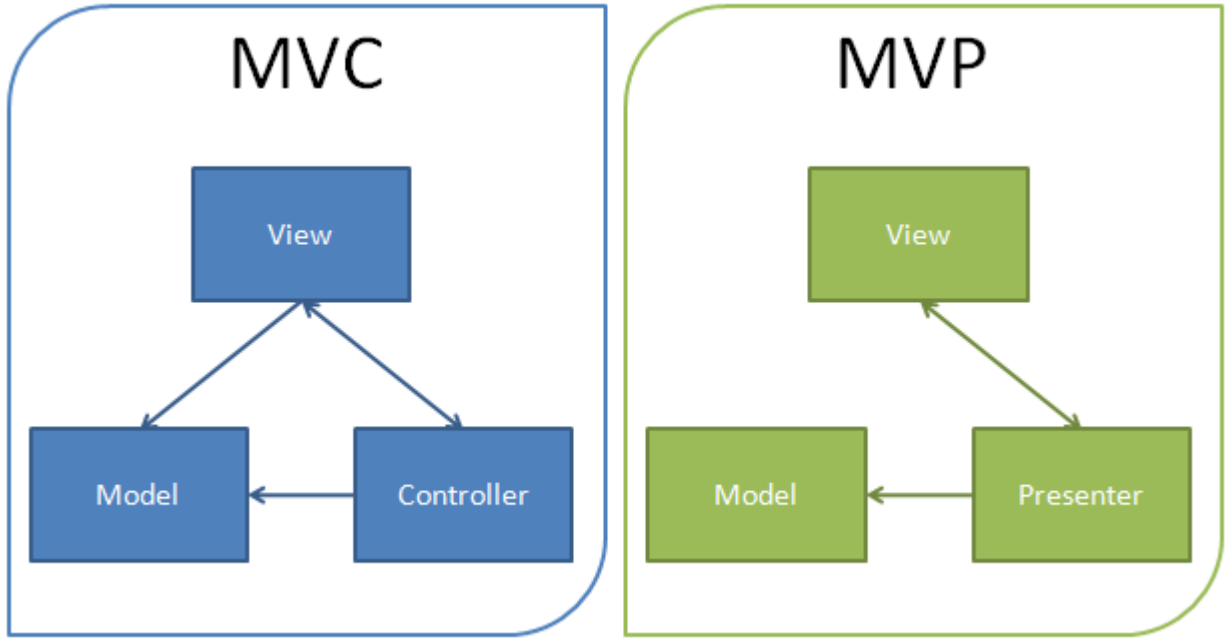
•**Coste de implantación alto:** una arquitectura de microservicios puede suponer un alto coste de implantación debido a costes de infraestructura y pruebas distribuidas.

Patrones

Patterns

Ejemplos de patrones arquitectónicos incluyen los siguientes:

- Programación por capas
- Tres niveles
- Pipeline
- Invocación implícita
- Arquitectura en pizarra
- Arquitectura dirigida por eventos, Presentación-abstracción-control
- Peer-to-peer
- Arquitectura orientada a servicios
- Objetos desnudos
- Modelo Vista Controlador



Fuente: https://ingsof2.wordpress.com/2016/04/28/modelo-vista-presentador/

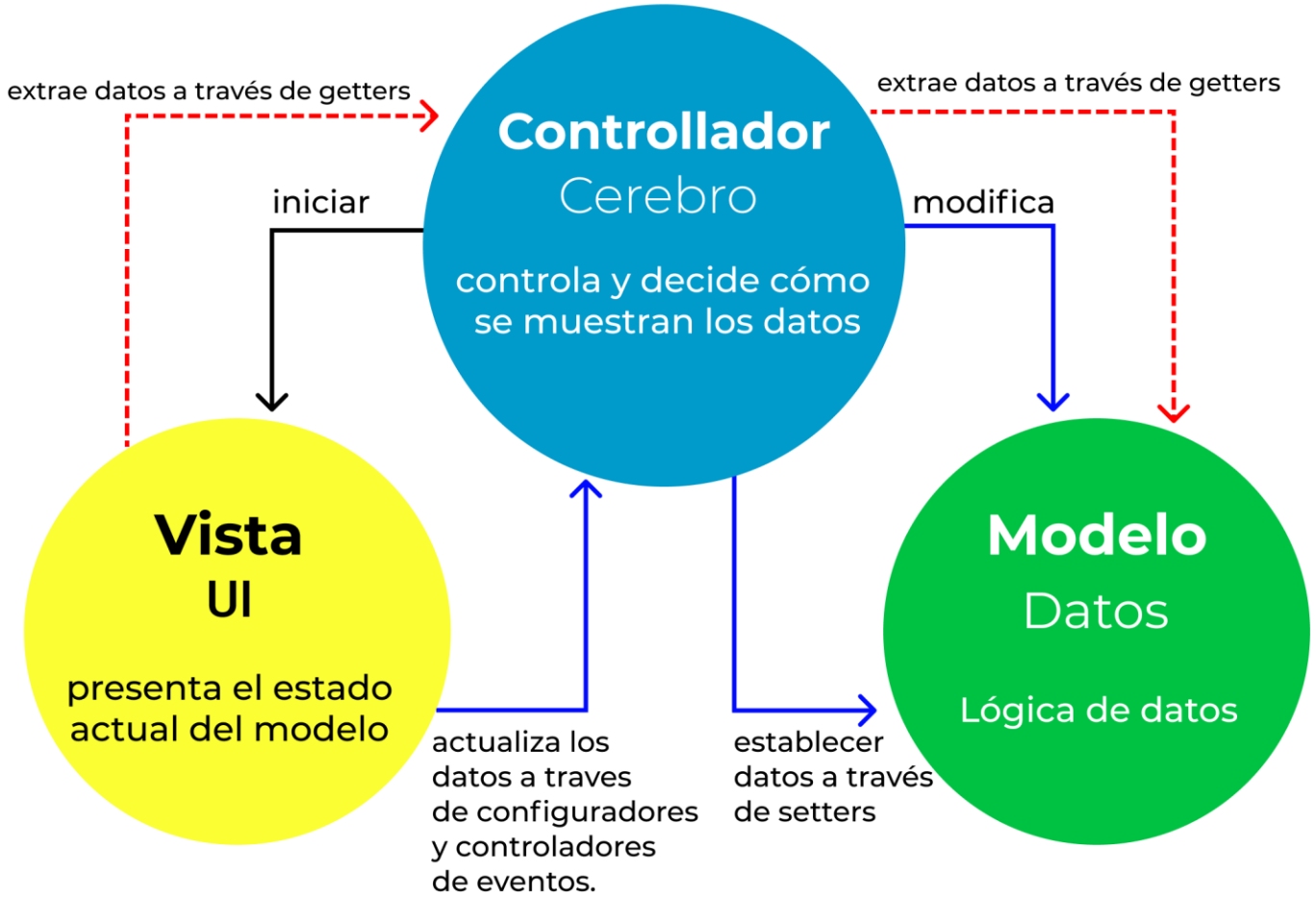
Patrones

Patterns

Los **patrones arquitectónicos**, o **patrones de arquitectura**, también llamados arquetipos ofrecen soluciones a problemas de arquitectura de software en ingeniería de software. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados. Un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones. En comparación con los patrones de diseño, los patrones arquitectónicos tienen un nivel de abstracción mayor.

- Patrones de arquitectura: Aquellos que expresan un esquema organizativo estructural fundamental para sistemas de software.
- Patrones de diseño: Aquellos que expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software.

Patrones de Arquitectura MVC



Fuente: https://hugojose14.medium.com/ruta-para-backend-developer-java-e606762ef350

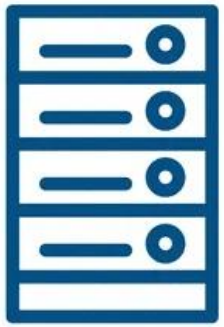
Fuente: https://diegochavez-dc.medium.com/patrones-de-dise%C3%B1o-de-software-2d3f14a503c5

Fuente: https://co.pinterest.com/pin/288582288627547019/

Roles en Desarrollo

Front End

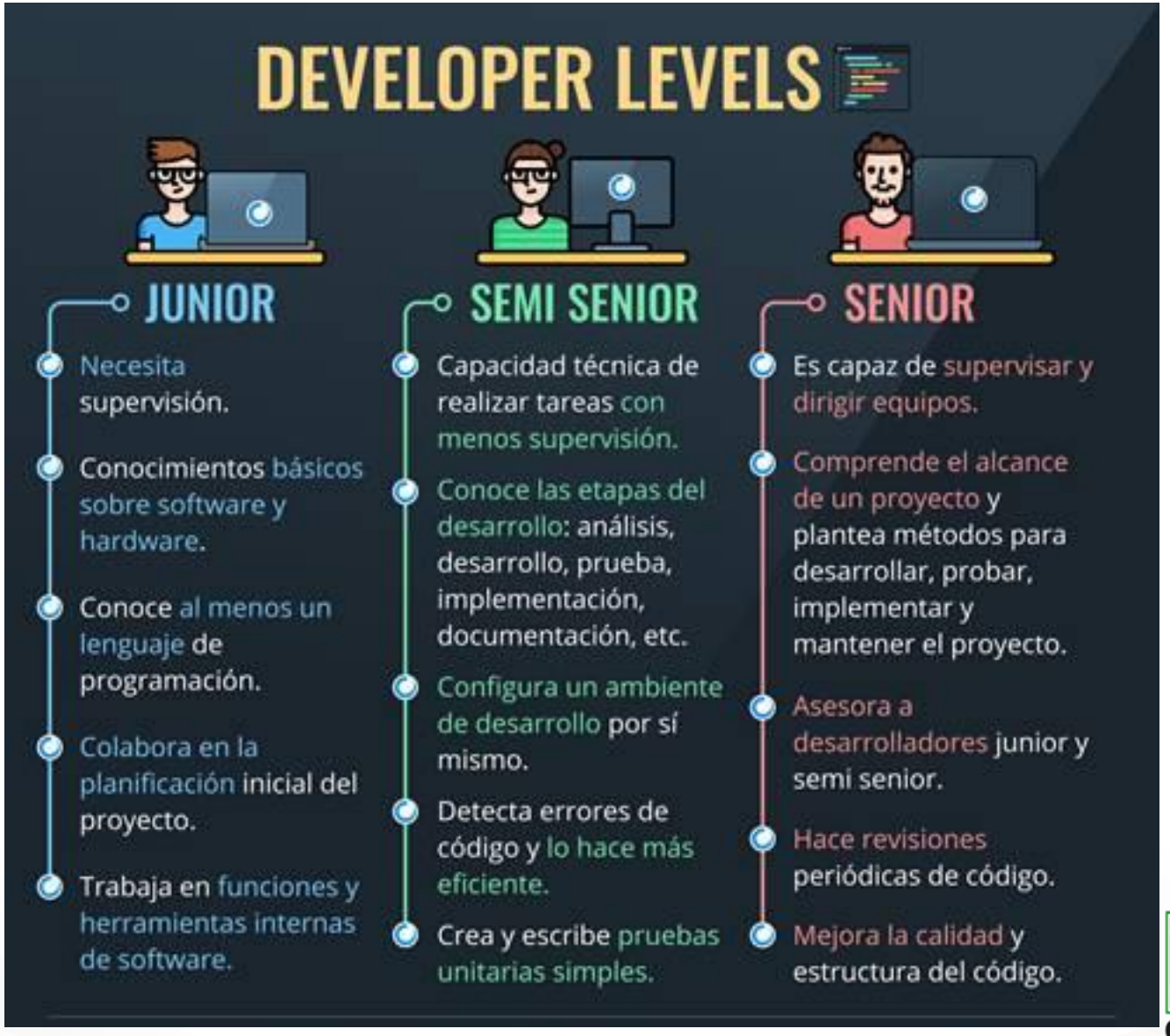
- Markup and web languages such as HTML, CSS and Javascript
- Asynchronous requests and Ajax
- Specialized web editing software
- Image editing
- Accessibility
- Cross-browser issues
- Search engine optimisation



Back End

- Programming and scripting such as Python, Ruby and/or Perl
- Server architecture
- Database administration
- Scalability
- Security
- Data transformation
- Backup

Fuente: https://kinsta.com/es/blog/desarrollador-de-backend/



Fuente: https://www.tiktok.com/discover/liceo-en-computaci%C3%B3n-css

Herramientas

Fuente: https://www.campusmvp.es/recursos/image.axd?picture=/2021/1T/lenguajes-programacion-back-end.png

Fuente: https://cursosdedesarrollo.com/wp-content/uploads/2020/06/backend-frameworks.png

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA

CORHUILA

CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

GRACIAS

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA

01 - Introducción a TypeScript

Ing. Julián Quimbayo

Ingeniería de Sistemas

Facultad de Ingeniería

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"

Stack para el Desarrollo de Software

Stack for Software Development

Node v18 o superior

Fuente: https://nodejs.org/es

Visual Studio Code

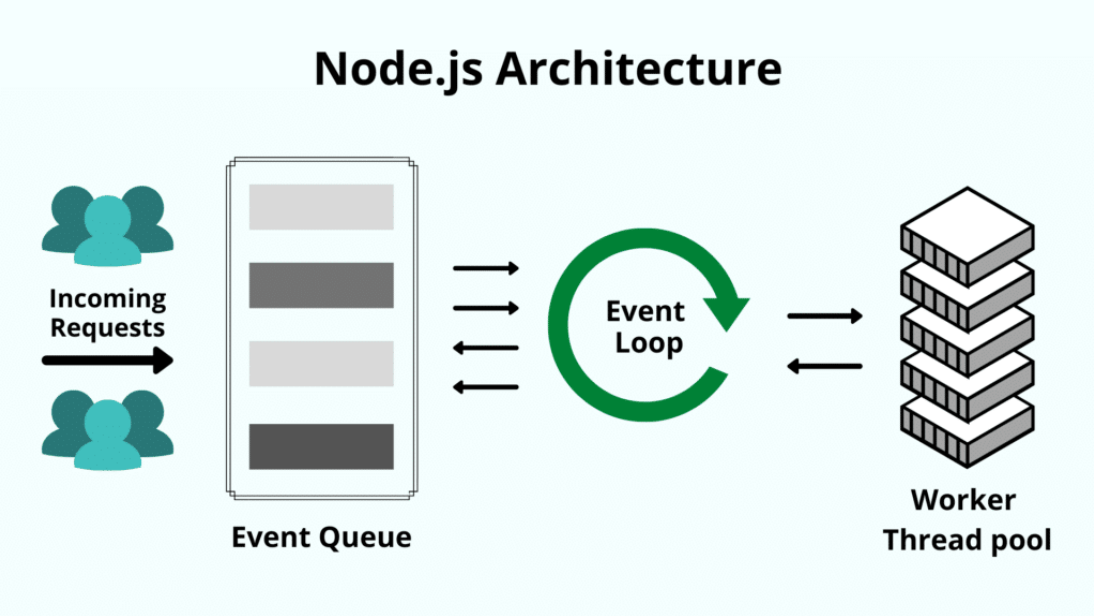
Fuente: https://code.visualstudio.com/

Manejador de Dependencias

Fuente: https://www.npmjs.com/

¿Qué es Node?

¿What is Node?



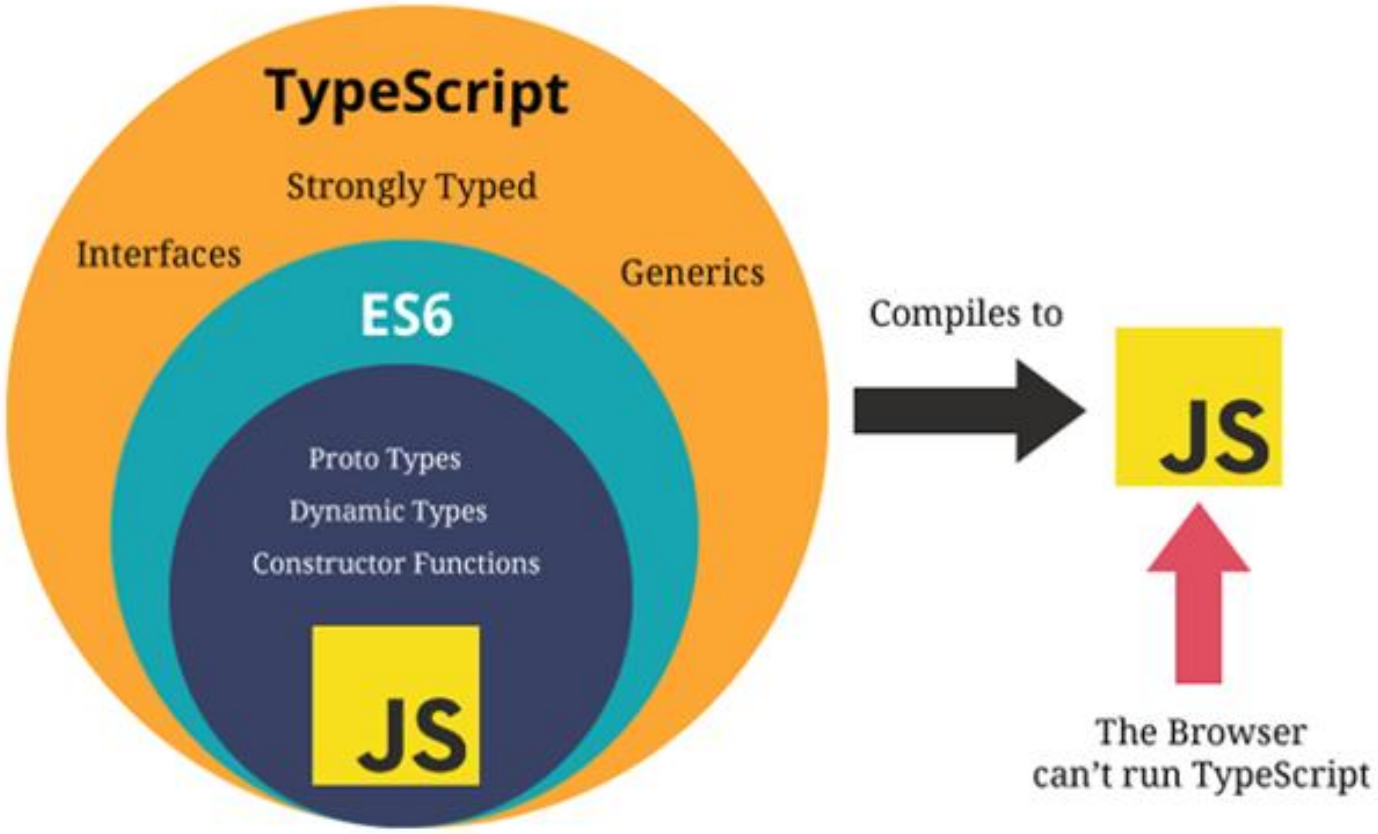
Fuente: https://co.pinterest.com/pin/what-is-nodejs-and-why-you-should-use-it--10135210911147990 62/

Aquí está la explicación de los componentes clave de la arquitectura que se ven en la imagen:

1. **Incoming Requests** (Solicitudes entrantes): Estas son las peticiones que llegan al servidor Node.js desde los clientes. Estas solicitudes pueden ser de distintos tipos, como peticiones HTTP, solicitudes de acceso a una base de datos, o cualquier otra tarea que requiera procesamiento en el servidor.
2. **Event Queue** (Cola de Eventos): Node.js utiliza una arquitectura basada en eventos, donde las solicitudes entrantes se colocan en una cola de eventos. Esta cola gestiona las solicitudes y las organiza para su procesamiento.
3. **Event Loop** (Bucle de Eventos): El bucle de eventos es el corazón de Node.js. Es un ciclo que revisa la cola de eventos de manera continua. Cuando hay una tarea en la cola de eventos, el bucle de eventos la toma y decide si puede ser procesada de inmediato (por ejemplo, operaciones que no bloquean) o si necesita pasarla a un hilo de trabajo.
4. **Worker Thread Pool** (Grupo de Hilos de Trabajo): Algunas tareas que llegan al servidor, como operaciones de I/O intensivas (por ejemplo, leer/escribir archivos, consultas a bases de datos) o cálculos complejos, no se pueden ejecutar directamente en el hilo principal porque bloquearían el bucle de eventos. Estas tareas se delegan a un grupo de hilos de trabajo en segundo plano, lo que permite que el bucle de eventos siga procesando otras solicitudes mientras se realiza el trabajo en segundo plano.

¿Para qué sirve TypeScript?

¿what is typsecript for?



Fuente: https://www.azulweb.net/typescript-la-evolucion-de-javascript/

TypeScript lo que hace es encapsular varios elementos:

- Una serie de funcionalidades de JavaScript 5, que es considerada hoy en día como un estándar y es la que todos los navegadores comprenden, incluso Node.js a nivel de servidor lo comprende.
- Como JavaScript ha seguido creciendo y apareció ECMAScript 6 (ES6), que añade nuevas funcionalidades, y que también está englobado dentro de TypeScript.
- Añade el uso un tipado muy estricto, en lugar del tipado dinámico de JavaScript, para intentar solucionar una serie de problemas. Aunque estos tipados no son requeridos, es recomendable utilizarlos para tener un código mucho más limpio.
- Añade la posibilidad de poder utilizar interfaces, para poder definir nuestros propios tipos o intentar aplicar programación orientada objetos luego.
- También añade la funcionalidad de genérica, que permite poder definir funciones que sean reutilizables, independientemente del tipo de datos que vayamos a tratar.

Creación de un Proyecto

Create a project

```
C:\Users\DELL\Desktop\IntroTypeScript>yarn create vite
yarn create v1.22.22
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
[4/4] Building fresh packages...

success Installed "create-vite@6.2.0" with binaries:
      - create-vite
      - cva
√ Project name: ... 01TypeScript-intro
√ Package name: ... 01typescript-intro
√ Select a framework: » Vanilla
√ Select a variant: » TypeScript
```

2

3

Fuente: Ing de Sistemas - Propia

Creación de un Proyecto

Create a project

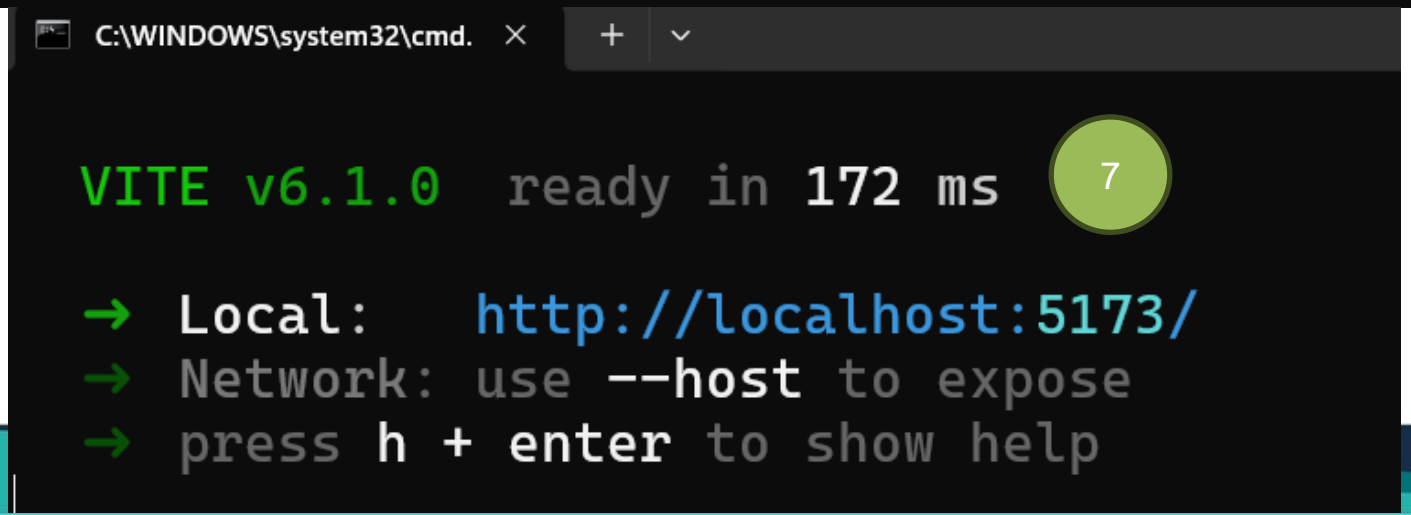
```
C:\Users\DELL\Desktop\IntroTypeScript\01TypeScript-intro>code .

C:\Users\DELL\Desktop\IntroTypeScript\01TypeScript-intro>yarn
yarn install v1.22.22
info No lockfile found.
[1/4] Resolving packages...
[2/4] Fetching packages...
info There appears to be trouble with your network connection. Retrying...
[##################################################-] 53/54
```

4

5

6

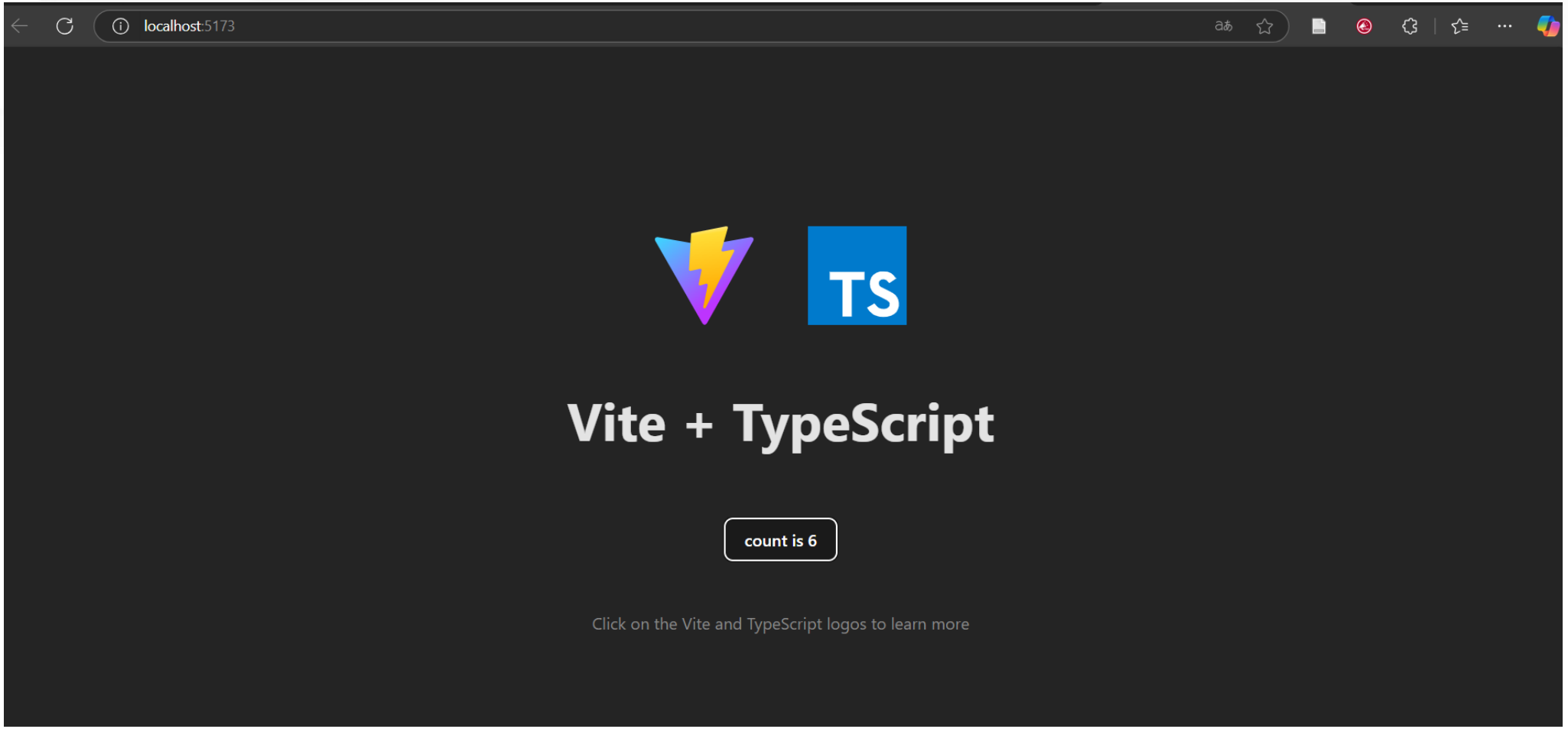
En la misma consola del CMD donde está el Proyecto digitamos: yarn dev y enter



Fuente: Ing de Sistemas - Propia

Creación de un Proyecto

Create a project



Fuente: Ing de Sistemas - Propia

CORHUILA

CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

GRACIAS

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA

02 - Manejo de Clases Typescript

Ing. Julián Quimbayo

Ingeniería de Sistemas

Facultad de Ingeniería

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"

Paso 1: Creación de Clase y Método Constructor

```
export class Pokemon{

    getImageUrl():string{
        return `https://pokemon.com/${this.id}.jpg`
    }

    constructor(
        public readonly id:number,
        public name:string,
       // public imageUrl:string
    ){}
```

Fuente: Ing de Sistemas - Propia

Esta clase Pokémon es como una ficha de personaje que guarda:

- *Un número único (ID) que no se puede cambiar*
- *El nombre del Pokémon*
- *Una función que nos da el link de su foto usando su ID*

Es similar a tener una tarjeta coleccionable donde cada Pokémon tiene su número y nombre, y podemos encontrar su imagen en internet usando ese número.

Paso 2: Creación de Clase y Método Constructor

```
//Métodos particulares - interpolación de String
private scream(){
    console.log(`${this.name.toUpperCase()}!!!!`)
}

speak(){
    console.log(`${this.name}, ${this.name}`)
    this.scream()
}
```

Fuente: Ing de Sistemas - Propia

Métodos de Texto en TypeScript Este código tiene dos funciones simples:

speak(): Hace que un personaje "hable" su nombre dos veces normalmente
scream(): Hace que el personaje "grite" su nombre en MAYÚSCULAS
Por ejemplo, si el nombre es "Pikachu":speak() mostraría: "Pikachu, Pikachu“ Y luego scream() mostraría: "PIKACHU!!!!“ Es como dar voz a un personaje en un juego o aplicación.

Paso 3: Creación de Clase y Método Constructor

```
async getMoves(){
    const{data} = await axios.get('https://pokeapi.co/api/v2/pokemon/1')

    console.log(data.moves)

    return data.moves;
}
```

Fuente: Ing de Sistemas - Propia

Función getMoves() - API Pokémon

Esta función hace 3 cosas simples:

1.Conecta con la API de Pokémon para obtener datos del Pokémon #1

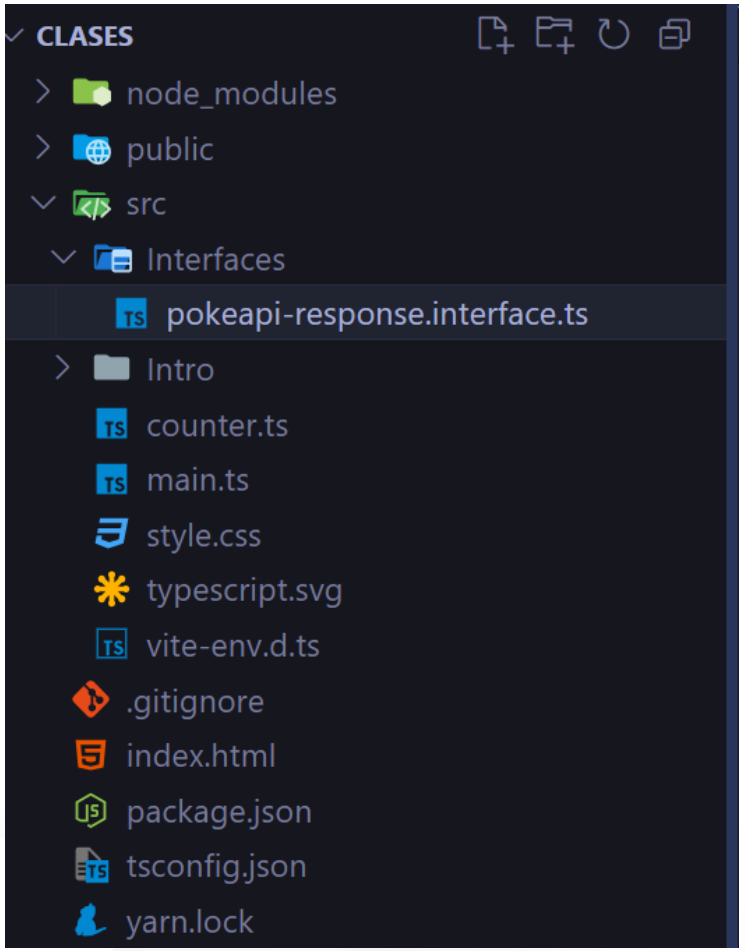
2.Extrae la información de sus movimientos

3.Muestra y devuelve la lista de movimientos

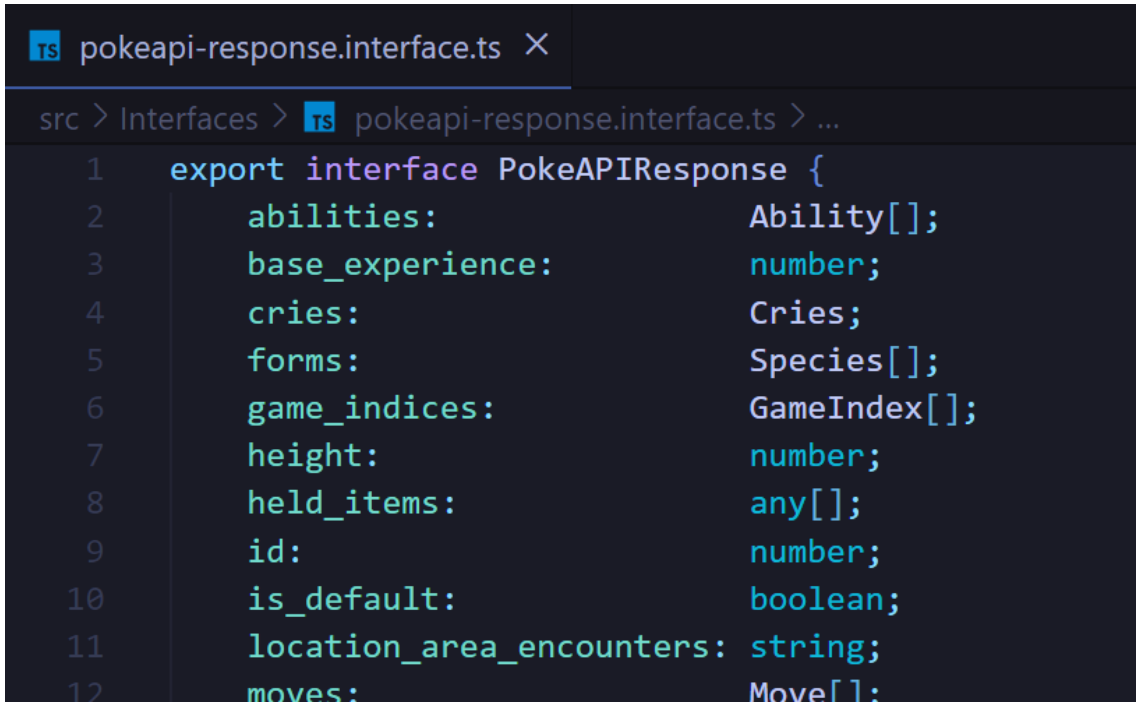
Es como pedirle a una base de datos: "Dame todos los ataques que conoce Bulbasaur" y esperar la respuesta.

Nota técnica: usa async/await para manejar la espera de datos

Paso 4: Interfaces



Fuente: Ing de Sistemas - Propia



Fuente: Ing de Sistemas - Propia

1.Se crea la carpeta Interfaces luego ahí mismo se construye el archivo denominado pokeapi-response.interface.ts

2.Luego se importa la interface dentro de la clase principal.

Paso 5: Interfaces

```
async getMoves(){
    const{data} = await axios.get<PokeAPIResponse>('https://pokeapi.co/api/v2/pokemon/1')

    console.log(data.)

    return data.moves
}


ort const bulbasur = n

onsole.log(bulbasur)

ulbasur.scream()

ulbasur.speak()
```

```
abilities        (property) PokeAPIResponse.abilities: Abi…
base_experience
cries
forms
game_indices
height
held_items
id
is_default
location_area_encounters
moves
name
```

Fuente: Ing de Sistemas - Propia

Como se pueden dar cuenta ya tiene accesos a cada uno de los encabezados del JSON.

En la segunda imagen se coloca la Promise con el objetivo que retorne los datos en formato de arreglo de Move que es el encabezados del movimiento del pokemon.

```
async getMoves():Promise<Move[]>{
    const{data} = await axios.get<PokeAPIResponse>('https://pokeapi.co/api/v2/pokemon/1')

    console.log(data.moves)

    return data.moves;
}
}
```

Fuente: Ing de Sistemas - Propia

Paso 6: Inyección de dependencias

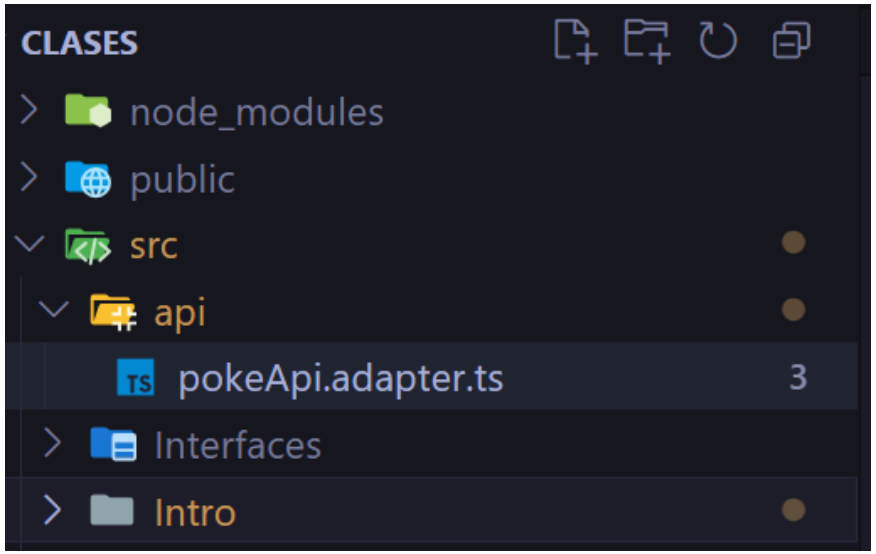
pokeApi.adapter.ts 3

src > api > pokeApi.adapter.ts > PokeApiAdapter > post

```ts
import axios from "axios";


export class PokeApiAdapter{
 private readonly axios=axios;

 async get (url: string){
    const {data} = await this.axios.get(url)
    return(data)
 }

 async patch (url: string){

 }

 async delete (url: string){

 }

 async post (url: string){

 }


}
```

Fuente: Ing de Sistemas - Propia

Permite entregar las herramientas en vez de que comience a buscar lo que necesita para el consumo del API.



Fuente: Ing de Sistemas - Propia

Paso 6: Inyección de dependencias

Se modifica el constructor en la clase de Injection.ts

1

```
constructor(
    public readonly id: number,
    public name: string,
    // Todo: inyectar dependencias
    private readonly http:PokeApiAdapter

) {}
```

Fuente: Ing de Sistemas - Propia

Se modifica la promise donde se recibe de tipo await pero acá se llama el this.http.get o cualquier método de la clase adapter.

Finalmente, se instancia la clase Adapter para pasar el parámetro en la petición.

2

```
    async getMoves(): Promise<Move[]> {
        //const { data } = await axios.get<PokeAPIResponse>('https://pokeapi.co/api/v2/pokemon/1');
        const data = await this.http.get('https://pokeapi.co/api/v2/pokemon/1')
        console.log( data.moves );

        return data.moves;
    }

}

const pokeApi = new PokeApiAdapter();

export const bulbasur = new Pokemon( 4, 'bulbasur', pokeApi );
```

Paso 7: Comienzo del Proyecto

1

Julian quiere poder establecer un sitio de Ecommerce donde se pueda realizar el proceso de compra y muestra de productos.

2

Crear un proyecto de Node que permita exponer los datos y los productos para que sea consumido desde la parte del Frontend.

NEST

3

Servicios

Autenticación

Seguridad

Fuente: Iconos creados por Pixel perfect from www.flaticon.com

Paso 8: ¿Por qué utilizar Nest?

¿Por qué NestJS?
Ventajas Principales

- **Arquitectura Robusta**: Estructura modular inspirada en Angular
- **TypeScript por defecto**: Código más seguro y mantenible
- **Decoradores y Módulos**: Código más limpio y organizado
- **Inyección de Dependencias**: Integrada de forma nativa

Comparado con otros

- Express: Más simple pero menos estructurado
- Koa: Más ligero, pero menos funcionalidades
- Fastify: Más rápido, pero menos ecosistema

Ideal para

- Proyectos empresariales grandes
- APIs escalables
- Equipos que vienen de Angular
- Microservicios

NEST

Fuente: Iconos creados por Pixel perfect from www.flaticon.com

Paso 9: Instalando Nest

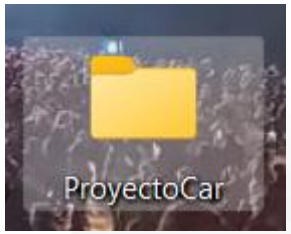
Ejecutamos el cmd como administrador y dentro digitamos el siguiente comando:

```
C:\Windows\System32>npm i -g @nestjs/cli
```

Comprobamos versión de Nest con:

```
C:\Windows\System32>nest --version
10.4.9
```

Creamos una carpeta con el nombre del proyecto: ProyectoCar y la abrimos con CMD, para crear el proyecto se hace con nest new ProyectoCar y enter, finalmente seleccionamos yarn como manejador de dependencias




```
C:\WINDOWS\system32\cmd.
Microsoft Windows [Versión 10.0.26100.3194]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\DELL\Desktop\ProyectoCar>nest new ProyectoCar
⚡  We will scaffold your app in a few seconds..

? Which package manager would you ❤️  to use?
  npm
> yarn
  pnpm
```

Fuente: Ing de Sistemas - Propia

NEST

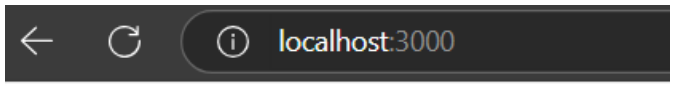
Fuente: Iconos creados por Pixel perfect from www.flaticon.com

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

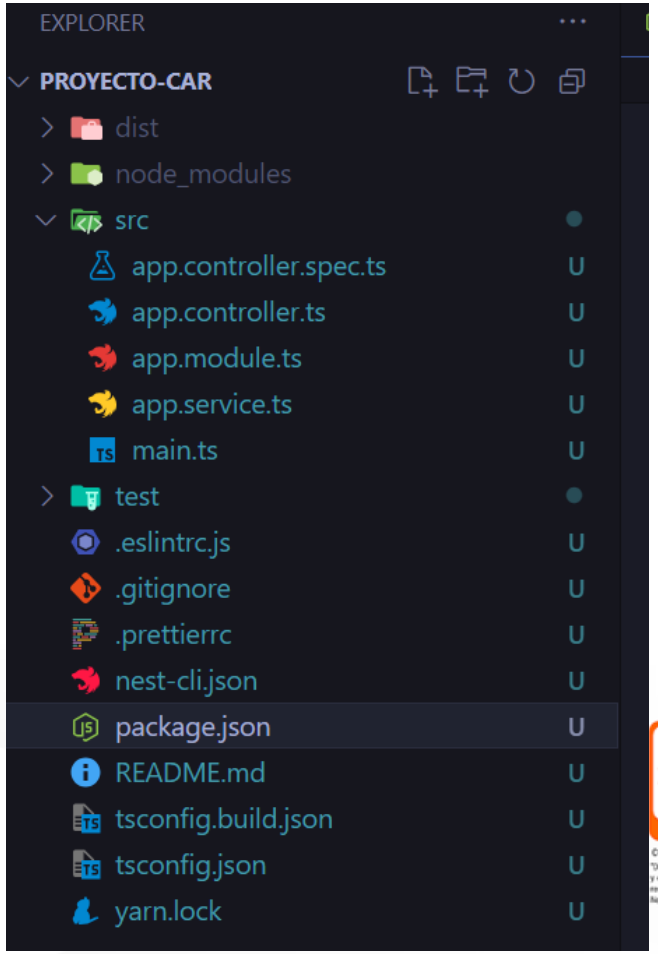
Paso 10: Lanzando el Proyecto

```
C:\WINDOWS\system32\cmd.

√ Installation in progress... ☕

🚀  Successfully created project proyecto-car
👉  Get started with the following commands:

$ cd proyecto-car
$ yarn run start


                          Thanks for installing Nest 🙏
                 Please consider donating to our open collective
                        to help us maintain this package.


                 🍷 Donate: https://opencollective.com/nest

C:\Users\DELL\Desktop\ProyectoCar>
```

```
C:\WINDOWS\system32\cmd.
                 🍷 Donate: https://opencollective.com/nest

C:\Users\DELL\Desktop\ProyectoCar>cd proyecto-car

C:\Users\DELL\Desktop\ProyectoCar\proyecto-car>yarn run start
yarn run v1.22.22
$ nest start
[Nest] 13380  - 20/02/2025, 7:19:42 p. m.     LOG [NestFactory] Starting Nest ap
plication...
[Nest] 13380  - 20/02/2025, 7:19:42 p. m.     LOG [InstanceLoader] AppModule dep
endencies initialized +10ms
[Nest] 13380  - 20/02/2025, 7:19:42 p. m.     LOG [RoutesResolver] AppController
 {/}: +5ms
[Nest] 13380  - 20/02/2025, 7:19:42 p. m.     LOG [RouterExplorer] Mapped {/, GE
T} route +3ms
[Nest] 13380  - 20/02/2025, 7:19:42 p. m.     LOG [NestApplication] Nest applica
tion successfully started +2ms
```



Hello World!



Fuente: Ing de Sistemas - Propia

CORHUILA

CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

GRACIAS

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA

03 - Clases y Métodos Asíncronos

Ing. Julián Quimbayo

Ingeniería de Sistemas

Facultad de Ingeniería

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"

Paso 1: Creación de Clase y Método Constructor

```
export class Pokemon{

    getImageUrl():string{
        return `https://pokemon.com/${this.id}.jpg`
    }

    constructor(
        public readonly id:number,
        public name:string,
       // public imageUrl:string
    ){}
```

Fuente: Ing de Sistemas - Propia

Esta clase Pokémon es como una ficha de personaje que guarda:

- *Un número único (ID) que no se puede cambiar*
- *El nombre del Pokémon*
- *Una función que nos da el link de su foto usando su ID*

Es similar a tener una tarjeta coleccionable donde cada Pokémon tiene su número y nombre, y podemos encontrar su imagen en internet usando ese número.

Paso 2: Creación de Clase y Método Constructor

```
//Métodos particulares - interpolación de String
private scream(){
    console.log(`${this.name.toUpperCase()}!!!!`)
}

speak(){
    console.log(`${this.name}, ${this.name}`)
    this.scream()
}
```

Fuente: Ing de Sistemas - Propia

Métodos de Texto en TypeScript Este código tiene dos funciones simples:

speak(): Hace que un personaje "hable" su nombre dos veces normalmente
scream(): Hace que el personaje "grite" su nombre en MAYÚSCULAS
Por ejemplo, si el nombre es "Pikachu":speak() mostraría: "Pikachu, Pikachu“ Y luego scream() mostraría: "PIKACHU!!!!“ Es como dar voz a un personaje en un juego o aplicación.

Paso 3: Creación de Clase y Método Constructor

```
async getMoves(){
    const{data} = await axios.get('https://pokeapi.co/api/v2/pokemon/1')

    console.log(data.moves)

    return data.moves;
}
```

Fuente: Ing de Sistemas - Propia

Función getMoves() - API Pokémon

Esta función hace 3 cosas simples:

1.Conecta con la API de Pokémon para obtener datos del Pokémon #1

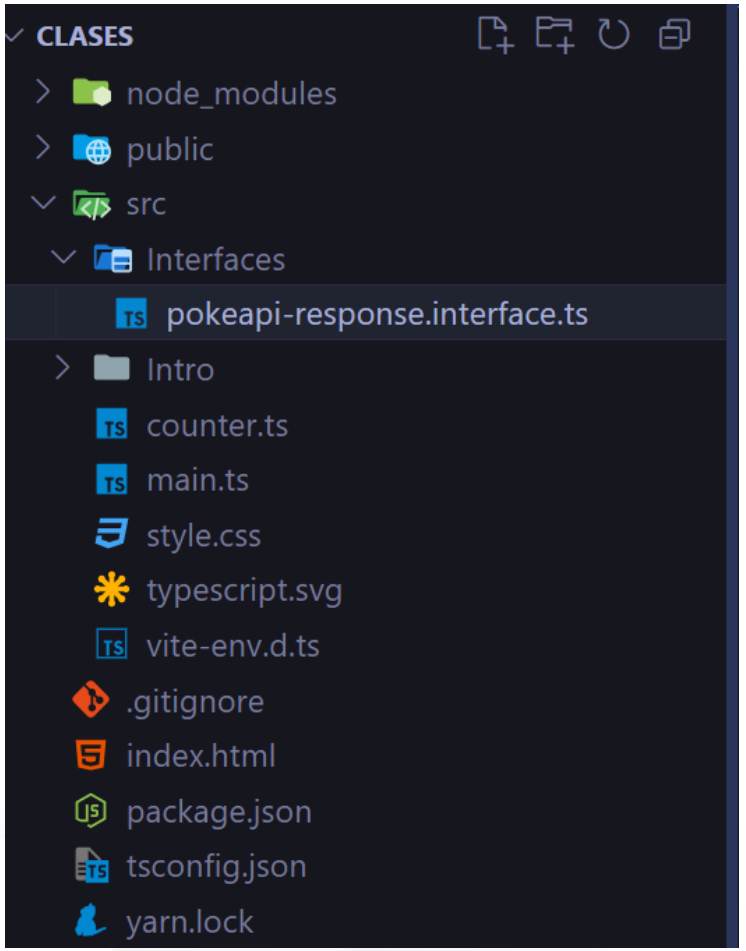
2.Extrae la información de sus movimientos

3.Muestra y devuelve la lista de movimientos

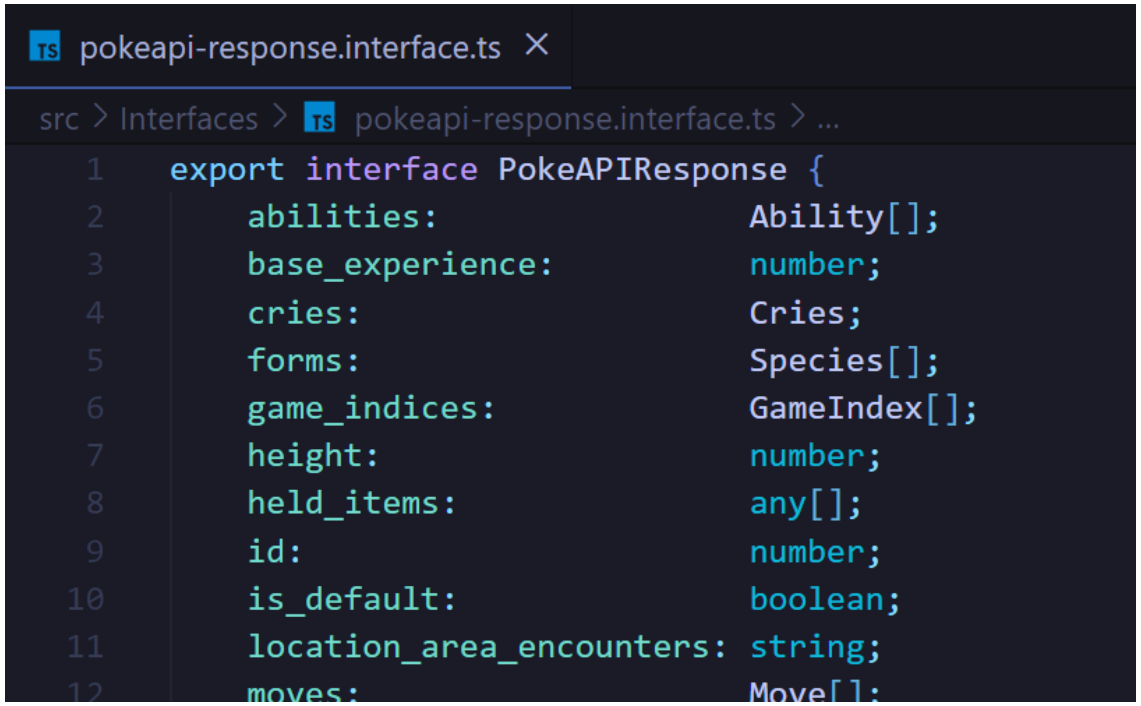
Es como pedirle a una base de datos: "Dame todos los ataques que conoce Bulbasaur" y esperar la respuesta.

Nota técnica: usa async/await para manejar la espera de datos

Paso 4: Interfaces



Fuente: Ing de Sistemas - Propia



Fuente: Ing de Sistemas - Propia

1.Se crea la carpeta Interfaces luego ahí mismo se construye el archivo denominado pokeapi-response.interface.ts

2.Luego se importa la interface dentro de la clase principal.

Paso 5: Interfaces

```
async getMoves(){
    const{data} = await axios.get<PokeAPIResponse>('https://pokeapi.co/api/v2/pokemon/1')

    console.log(data.)

    return data.moves
}


ort const bulbasur = n

onsole.log(bulbasur)

ulbasur.scream()

ulbasur.speak()
```

```
abilities        (property) PokeAPIResponse.abilities: Abi…
base_experience
cries
forms
game_indices
height
held_items
id
is_default
location_area_encounters
moves
name
```

Fuente: Ing de Sistemas - Propia

Como se pueden dar cuenta ya tiene accesos a cada uno de los encabezados del JSON.

En la segunda imagen se coloca la Promise con el objetivo que retorne los datos en formato de arreglo de Move que es el encabezados del movimiento del pokemon.

```
async getMoves():Promise<Move[]>{
    const{data} = await axios.get<PokeAPIResponse>('https://pokeapi.co/api/v2/pokemon/1')

    console.log(data.moves)

    return data.moves;
}
}
```

Fuente: Ing de Sistemas - Propia

Paso 6: Inyección de dependencias

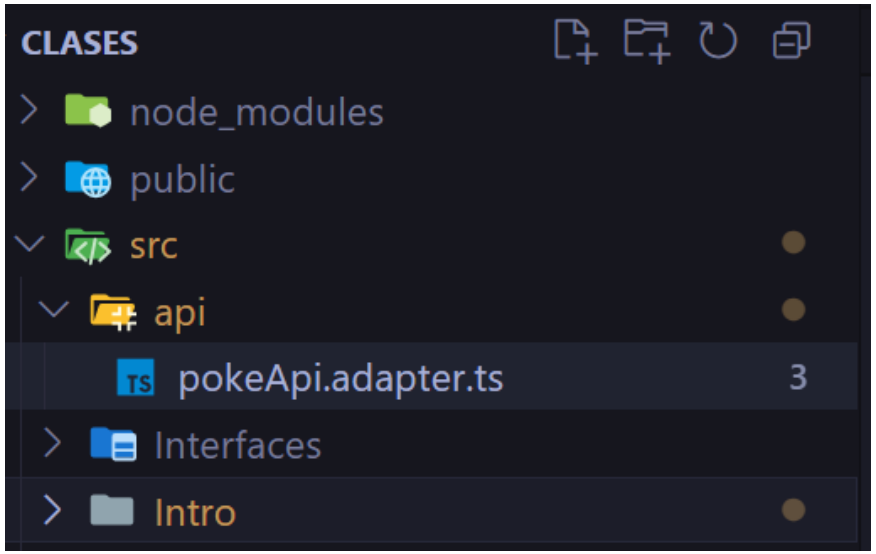
pokeApi.adapter.ts 3

src > api > pokeApi.adapter.ts > PokeApiAdapter > post

```ts
import axios from "axios";


export class PokeApiAdapter{
 private readonly axios=axios;

 async get (url: string){
    const {data} = await this.axios.get(url)
    return(data)
 }

 async patch (url: string){

 }

 async delete (url: string){

 }

 async post (url: string){

 }


}
```

Fuente: Ing de Sistemas - Propia

Permite entregar las herramientas en vez de que comience a buscar lo que necesita para el consumo del API.



Fuente: Ing de Sistemas - Propia

Paso 6: Inyección de dependencias

Se modifica el constructor en la clase de Injection.ts

1

```
constructor(
    public readonly id: number,
    public name: string,
    // Todo: inyectar dependencias
    private readonly http:PokeApiAdapter

) {}
```

Fuente: Ing de Sistemas - Propia

Se modifica la promise donde se recibe de tipo await pero acá se llama el this.http.get o cualquier método de la clase adapter.

Finalmente, se instancia la clase Adapter para pasar el parámetro en la petición.

2

```
    async getMoves(): Promise<Move[]> {
        //const { data } = await axios.get<PokeAPIResponse>('https://pokeapi.co/api/v2/pokemon/1');
        const data = await this.http.get('https://pokeapi.co/api/v2/pokemon/1')
        console.log( data.moves );

        return data.moves;
    }

}

const pokeApi = new PokeApiAdapter();

export const bulbasur = new Pokemon( 4, 'bulbasur', pokeApi );
```

Paso 7: Comienzo del Proyecto

1

Julian quiere poder establecer un sitio de Ecommerce donde se pueda realizar el proceso de compra y muestra de productos.

2

Crear un proyecto de Node que permita exponer los datos y los productos para que sea consumido desde la parte del Frontend.

NEST

3

Servicios

Autenticación

Seguridad

Fuente: Iconos creados por Pixel perfect from www.flaticon.com

Paso 8: ¿Por qué utilizar Nest?

¿Por qué NestJS?
Ventajas Principales

- **Arquitectura Robusta**: Estructura modular inspirada en Angular
- **TypeScript por defecto**: Código más seguro y mantenible
- **Decoradores y Módulos**: Código más limpio y organizado
- **Inyección de Dependencias**: Integrada de forma nativa

Comparado con otros

- Express: Más simple pero menos estructurado
- Koa: Más ligero, pero menos funcionalidades
- Fastify: Más rápido, pero menos ecosistema

Ideal para

- Proyectos empresariales grandes
- APIs escalables
- Equipos que vienen de Angular
- Microservicios

NEST

Fuente: Iconos creados por Pixel perfect from www.flaticon.com

Paso 9: Instalando Nest

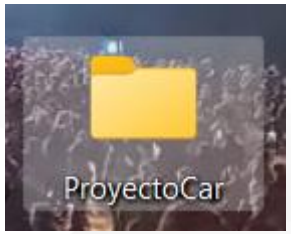
Ejecutamos el cmd como administrador y dentro digitamos el siguiente comando:

```
C:\Windows\System32>npm i -g @nestjs/cli
```

Comprobamos versión de Nest con:

```
C:\Windows\System32>nest --version
10.4.9
```

Creamos una carpeta con el nombre del proyecto: ProyectoCar y la abrimos con CMD, para crear el proyecto se hace con nest new ProyectoCar y enter, finalmente seleccionamos yarn como manejador de dependencias




```
C:\WINDOWS\system32\cmd.
Microsoft Windows [Versión 10.0.26100.3194]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\DELL\Desktop\ProyectoCar>nest new ProyectoCar
⚡  We will scaffold your app in a few seconds..

? Which package manager would you ❤️  to use?
  npm
> yarn
  pnpm
```

Fuente: Ing de Sistemas - Propia

NEST

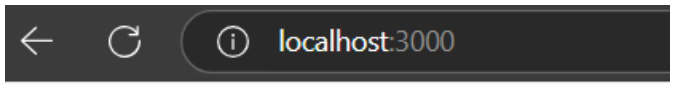
Fuente: Iconos creados por Pixel perfect from www.flaticon.com

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

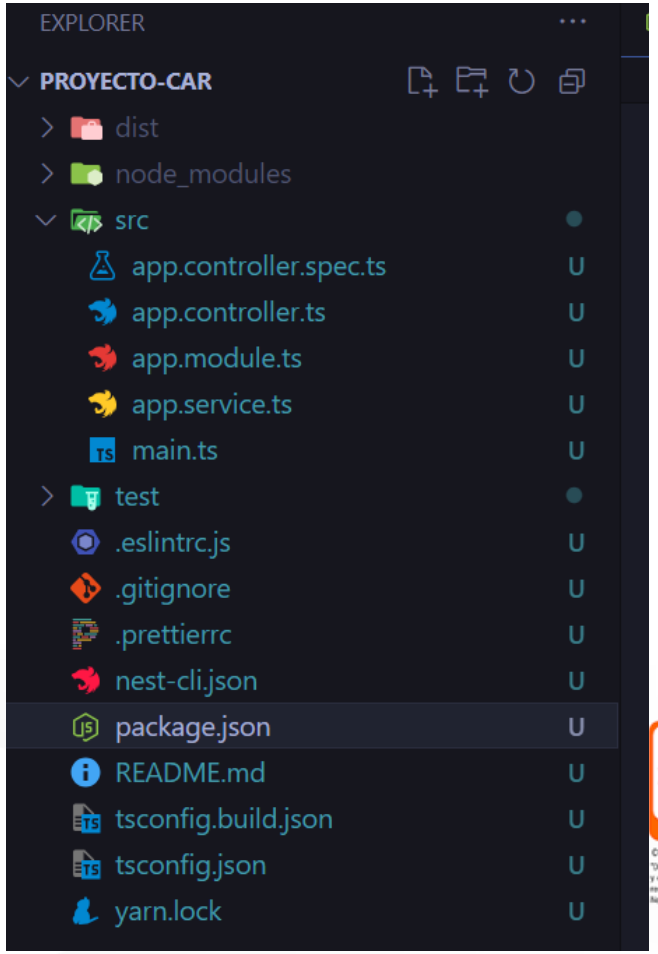
Paso 10: Lanzando el Proyecto

```
C:\WINDOWS\system32\cmd.

√ Installation in progress... ☕

🚀  Successfully created project proyecto-car
👉  Get started with the following commands:

$ cd proyecto-car
$ yarn run start


                          Thanks for installing Nest 🙏
                 Please consider donating to our open collective
                        to help us maintain this package.


                 🍷 Donate: https://opencollective.com/nest

C:\Users\DELL\Desktop\ProyectoCar>
```

```
C:\WINDOWS\system32\cmd.
                 🍷 Donate: https://opencollective.com/nest

C:\Users\DELL\Desktop\ProyectoCar>cd proyecto-car

C:\Users\DELL\Desktop\ProyectoCar\proyecto-car>yarn run start
yarn run v1.22.22
$ nest start
[Nest] 13380  - 20/02/2025, 7:19:42 p. m.     LOG [NestFactory] Starting Nest ap
plication...
[Nest] 13380  - 20/02/2025, 7:19:42 p. m.     LOG [InstanceLoader] AppModule dep
endencies initialized +10ms
[Nest] 13380  - 20/02/2025, 7:19:42 p. m.     LOG [RoutesResolver] AppController
 {/}: +5ms
[Nest] 13380  - 20/02/2025, 7:19:42 p. m.     LOG [RouterExplorer] Mapped {/, GE
T} route +3ms
[Nest] 13380  - 20/02/2025, 7:19:42 p. m.     LOG [NestApplication] Nest applica
tion successfully started +2ms
```



Hello World!



Fuente: Ing de Sistemas - Propia

CORHUILA

CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

GRACIAS

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA

04 - Comenzando Proyecto en NEST

Ing. Julián Quimbayo

Ingeniería de Sistemas

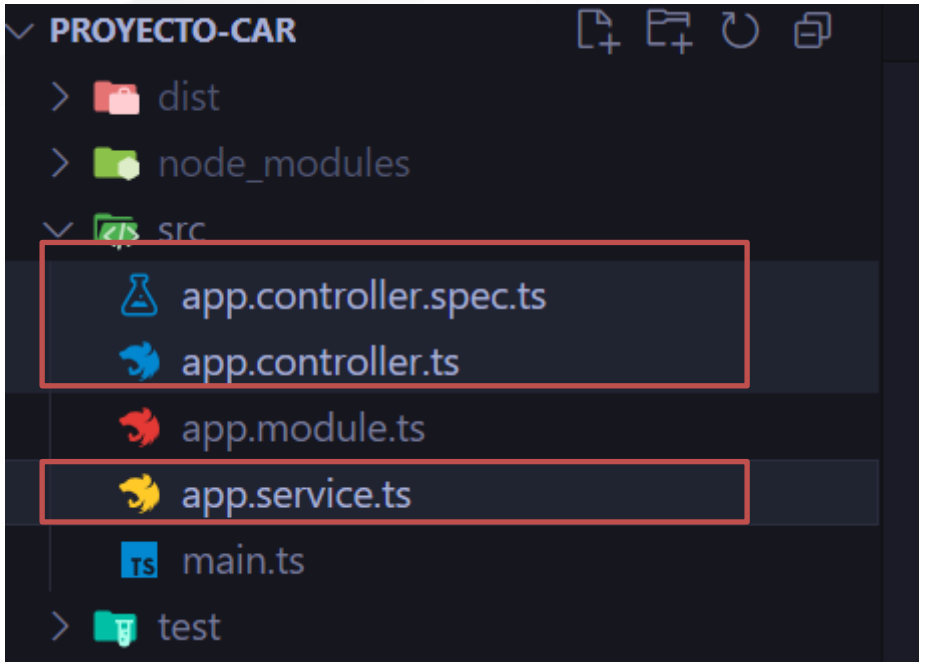
Facultad de Ingeniería

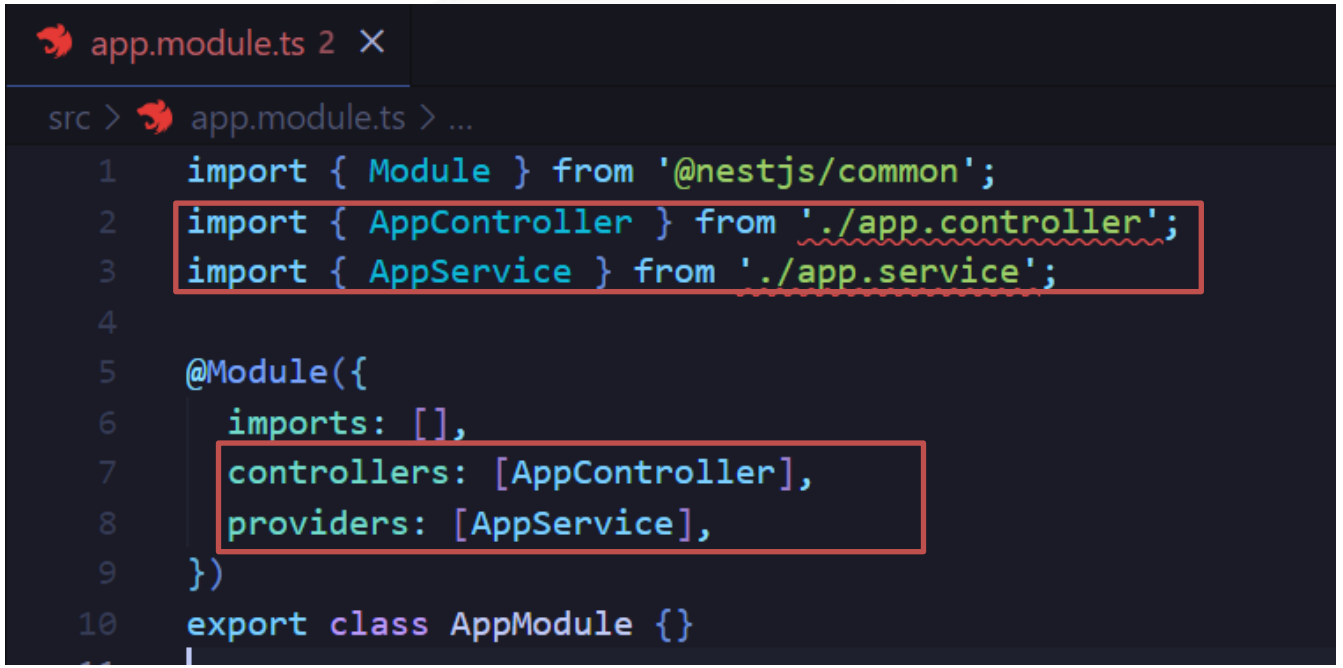
CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

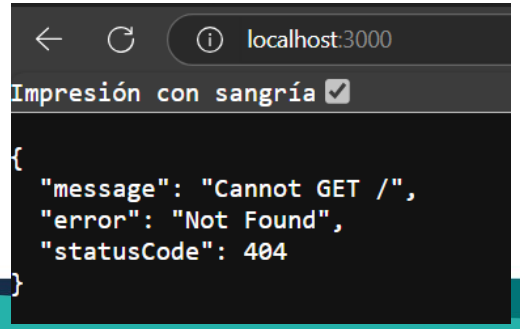
CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"

PASO 1: App.module.ts

Eliminamos los archivos y procedemos a levantar el servicio, recuerda probar la Api con Postman.







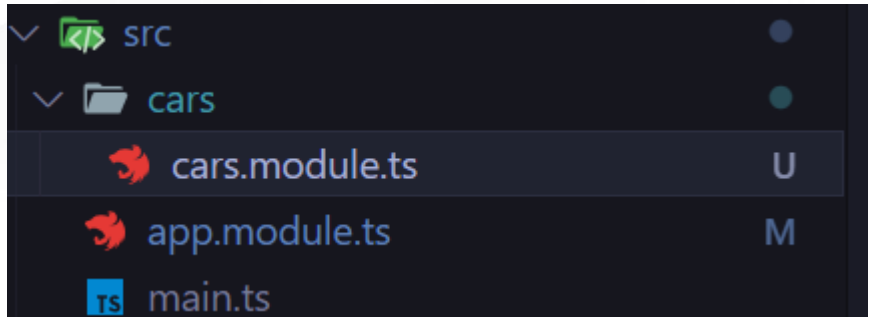
Desde cmd yarn run start:dev y enter

Fuente: Ing de Sistemas - Propia

PASO 2: Controladores

Creamos el módulo cars con ayuda del comando de Nest

```
PS C:\Users\DELL\Desktop\ProyectoCar\proyecto-car> nest g mo cars
CREATE src/cars/cars.module.ts (85 bytes)
UPDATE src/app.module.ts (215 bytes)
PS C:\Users\DELL\Desktop\ProyectoCar\proyecto-car>
```



Creamos el controlador cars con ayuda del comando de Nest

```
PS C:\Users\DELL\Desktop\ProyectoCar\proyecto-car> nest g co cars
CREATE src/cars/cars.controller.ts (101 bytes)
CREATE src/cars/cars.controller.spec.ts (496 bytes)
UPDATE src/cars/cars.module.ts (170 bytes)
```

Fuente: Ing de Sistemas - Propia

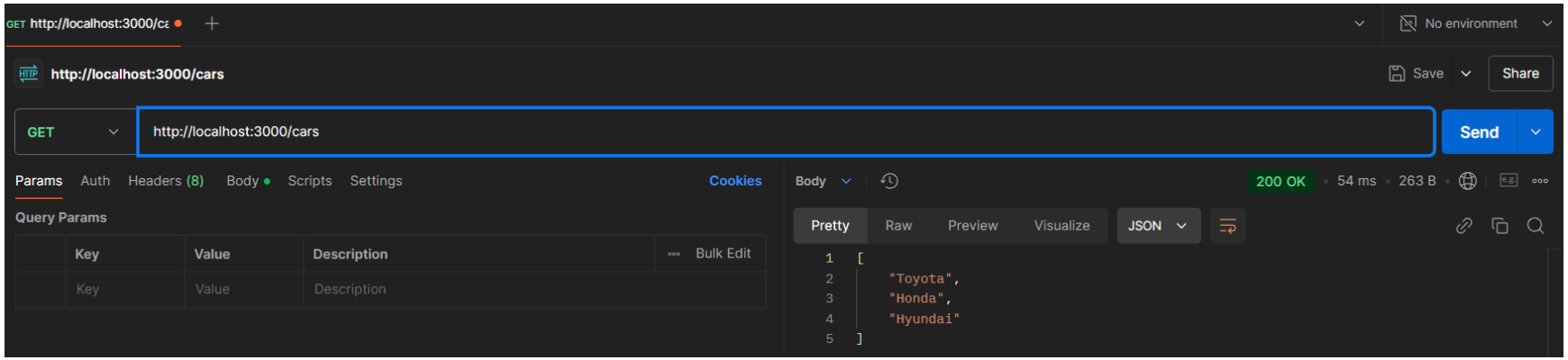
PASO 2: Controladores

Actualizamos el Controlador de Cars pasándole valores

app.module.ts M cars.controller.ts U

src > cars > cars.controller.ts > CarsController > getAllCars

```ts
import { Controller, Get } from '@nestjs/common';

@Controller('cars')
export class CarsController {
    @Get()
    getAllCars(){
        return ['Toyota','Honda','Hyundai']
    }
}
```



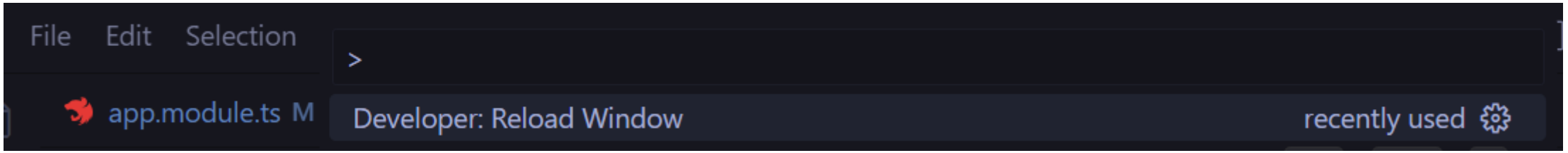
Fuente: Ing de Sistemas - Propia

PASO 3: Desactivando Prettier

En el cmd ejecutamos yarn remove prettier y enter, esto desinstala de las dependencias prettier de Nest.

```
PS C:\Users\DELL\Desktop\ProyectoCar\proyecto-car> yarn remove prettier
yarn remove v1.22.22
[1/2] Removing module prettier...
[2/2] Regenerating lockfile and installing missing dependencies...
warning " > eslint-plugin-prettier@5.2.3" has unmet peer dependency "prettier@>=3.0.0".
warning " > ts-loader@9.5.2" has unmet peer dependency "webpack@^5.0.0".
success Uninstalled packages.
Done in 10.89s.
```

Buscamos con F1 en visual : developer: reload window damos click a la opción y carga el visual de cero



Fuente: Ing de Sistemas - Propia

PASO 4: Buscado datos Por ID

Modificamos el controlador creando una variable privada y asignando los valores. Usamos this para leerlo en el retorno.

app.module.ts M cars.controller.ts U

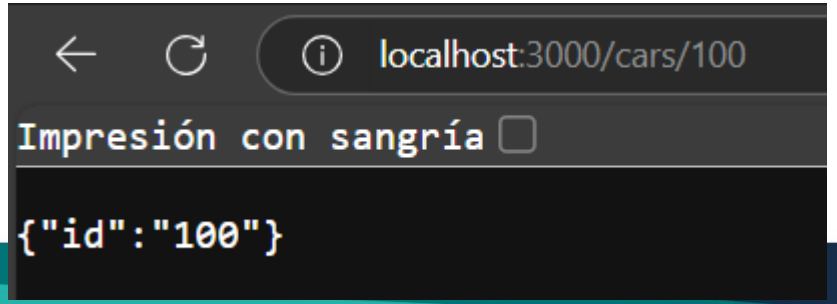
src > cars > cars.controller.ts > CarsController

```ts
import { Controller, Get } from '@nestjs/common';

@Controller('cars')
export class CarsController {

    private cars = ['Toyota','Honda','Hyundai']

    @Get()
    getAllCars(){
        return this.cars
    }
}
```

app.module.ts M cars.controller.ts U

src > cars > cars.controller.ts > CarsController > getCarById

```ts
export class CarsController {
    }

    @Get(':id')
    getCarById(@Param('id')id){
        console.log({id})
        return{
            id
        }
    }
}
```



Fuente: Ing de Sistemas - Propia

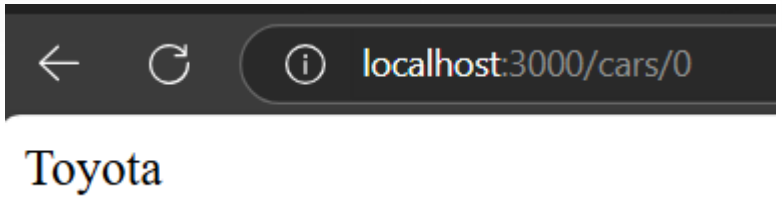
PASO 5: Buscado datos Por ID

Reto: Usando el arreglo definido en variable cars devuelve el valor que corresponde según el Id.

app.module.ts M cars.controller.ts U

src > cars > cars.controller.ts > CarsController > get(

```
4   export class CarsController {
9       getAllCars(){
10          return this.cars
11      }
12
13      @Get(':id')
14      getCarById(@Param('id')id){
15          console.log({id})
16          return this.cars[id]
17      }
18  }
```



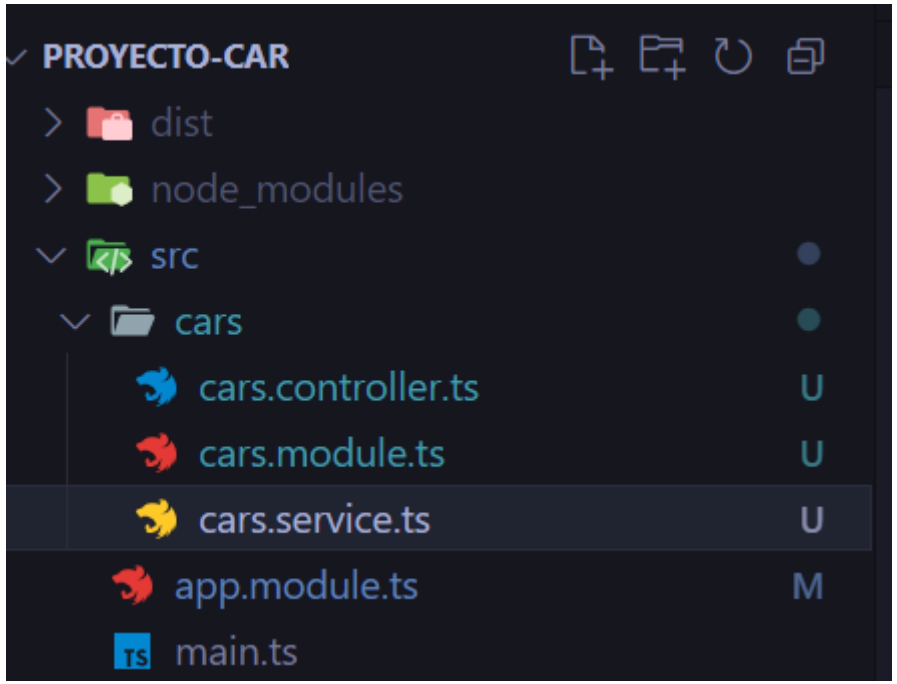
Si hacemos peticiones a un valor que no corresponde a un dato del arreglo aún así nos confirma el status 200 pero no muestra nada, esto se debe controlar con manejo de errores y enviar un status 404 por ejemplo.

Fuente: Ing de Sistemas - Propia

PASO 6: Creando un Servicio

Se crea el servicio usando el comando de Nest.

```
PS C:\Users\DELL\Desktop\ProyectoCar\proyecto-car> nest g s cars --no-spec
CREATE src/cars/cars.service.ts (92 bytes)
UPDATE src/cars/cars.module.ts (244 bytes)
```



Fuente: Ing de Sistemas - Propia

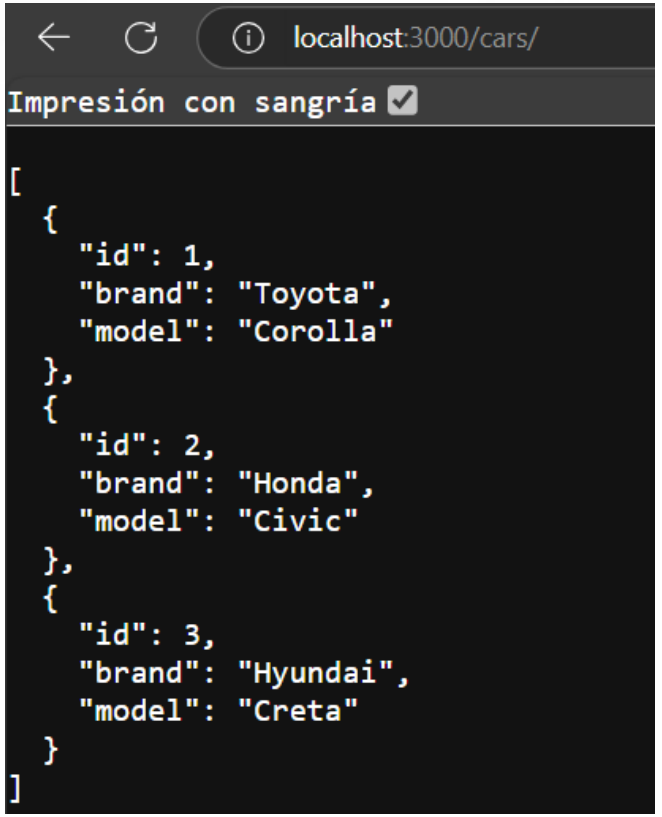
PASO 7: Inyección de Dependencias

Modificamos el servicio pasando los datos del controller a este archivo pero en un formato JSON.

```
@Injectable()
export class CarsService {
    private cars = [
        {
            id:1,
            brand: 'Toyota',
            model:'Corolla'
        },
        {
            id:2,
            brand: 'Honda',
            model:'Civic'
        },
        {
            id:3,
            brand: 'Hyundai',
            model:'Creta'
        }
    ]
}
```

```
import { Controller, Get, Param } from '@nestjs/common';
import { CarsService } from './cars.service';

@Controller('cars')
export class CarsController {

    constructor(
        private readonly carsService: CarsService
    ){}

    @Get()
    getAllCars(){
        return this.carsService.findAll()
    }
```

```
findAll(){

Return this.cars
    }
Agregar en el Service.
```



Fuente: Ing de Sistemas - Propia

PASO 8: Reto Inyección de Dependencias

Agregar el siguiente código a el servicio y hacer lo mismo del muestreo de todos los datos pero esta vez por un solo ID.

```
findOneById(id:number){

}
```

```
findOneById(id:number){
    const car = this.cars.find(car => car.id===id);

    return car;
}
```

app.module.ts M | cars.controller.ts U | cars.service.ts U

src > cars > cars.controller.ts > CarsController > getCarById

```
5   export class CarsController {
12      getAllCars(){
            return this.carsService.findAll()
14      }
15
16      @Get(':id')
17      getCarById(@Param('id')id){
18          console.log({id})
19          return this.carsService.findOneById(+id);
20      }
21  }
```

Fuente: Ing de Sistemas - Propia

CORHUILA

CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

GRACIAS

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA

05 – DTO y Validación de Información Final

Ing. Julián Quimbayo

Ingeniería de Sistemas

Facultad de Ingeniería

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA
"Diseño y prestación de servicios de docencia, investigación y extensión de programas de pregrado, aplicando todos los requisitos de las normas ISO implementadas en sus sedes Neiva y Pitalito"

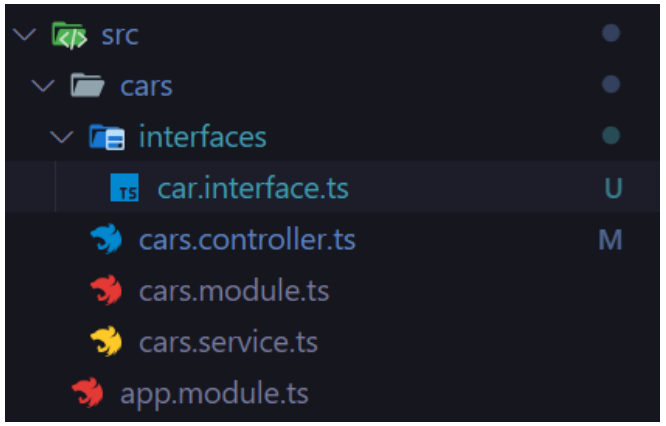
Paso 1: Montaje del Proyecto

1. GitHub - jaquimbayoc7/ProyectoNest at master

Clonar el proyecto de GitHub

```
# Clonar el repositorio
git clone https://github.com/jaquimbayoc7/ProyectoNest.git

# Navegar al directorio del proyecto
cd ProyectoNest

# Instalar dependencias con npm
npm install

# O si prefieres usar yarn
yarn install
```

Paso 2: Interfaces




```
export interface Car{
    id:number;
    model:string;
    brand:string;
}
```

car.interface.ts U | cars.service.ts M

src > cars > cars.service.ts > CarsService > cars

```
1  import { Injectable, NotFoundException } from '@nestjs/common';
2  import { Car } from './interfaces/car.interface';
3
4  @Injectable()
5  export class CarsService {
6    private cars: Car[] = [
```

Fuente: Ing de Sistemas - Propia

Paso 3: Implementar UUID

car.interface.ts U | cars.service.ts 4, M

src > cars > interfaces > car.interface.ts > Car

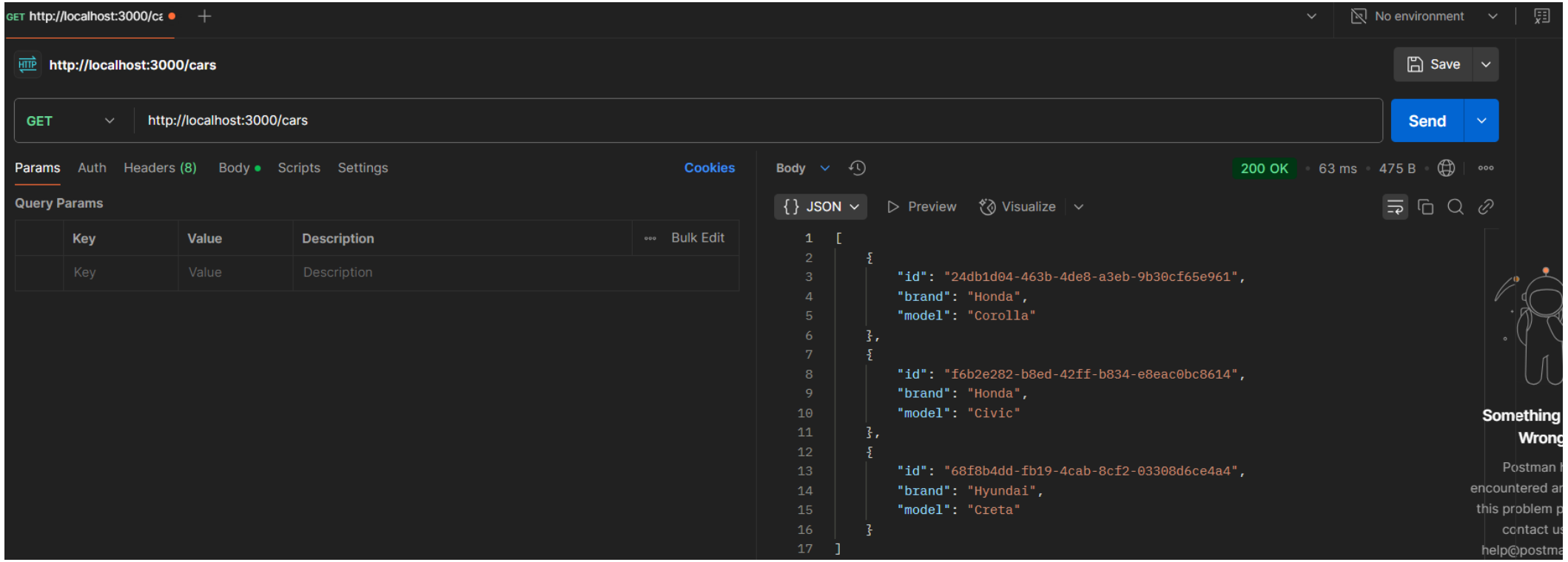
```ts
export interface Car{
    id:string;
    model:string;
    brand:string;
}
```

```
PS C:\Users\DELL\Desktop\ProyectoCar\proyecto-car> yarn add uuid
yarn add v1.22.22
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
warning " > eslint-plugin-prettier@5.2.3" has unmet peer dependency "prettier@>=3.0.0".
warning " > ts-loader@9.5.2" has unmet peer dependency "webpack@^5.0.0".
[4/4] Building fresh packages...

success Saved lockfile.
success Saved 1 new dependency.
info Direct dependencies
└─ uuid@11.1.0
info All dependencies
└─ uuid@11.1.0
Done in 6.71s.
PS C:\Users\DELL\Desktop\ProyectoCar\proyecto-car>
```

```
PS C:\Users\DELL\Desktop\ProyectoCar\proyecto-car> yarn add -D @types/uuid
yarn add v1.22.22
[1/4] Resolving packages...
[2/4] Fetching packages...
[3/4] Linking dependencies...
warning " > eslint-plugin-prettier@5.2.3" has unmet peer dependency "prettier@>=3.0.0".
warning " > ts-loader@9.5.2" has unmet peer dependency "webpack@^5.0.0".
[4/4] Building fresh packages...

success Saved lockfile.
success Saved 1 new dependency.
info Direct dependencies
└─ @types/uuid@10.0.0
info All dependencies
└─ @types/uuid@10.0.0
Done in 6.44s.
```

Fuente: Ing de Sistemas - Propia

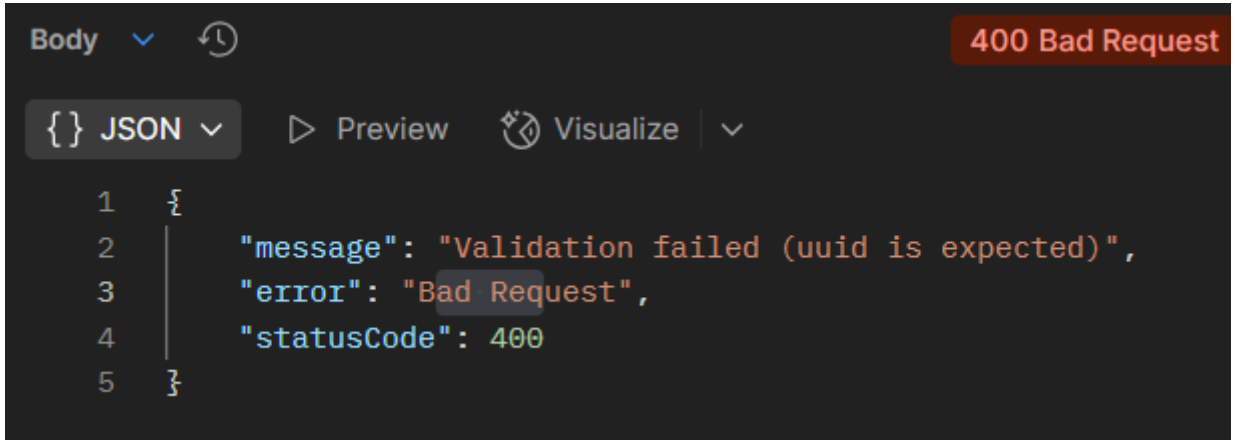
Paso 3: Implementar UUID

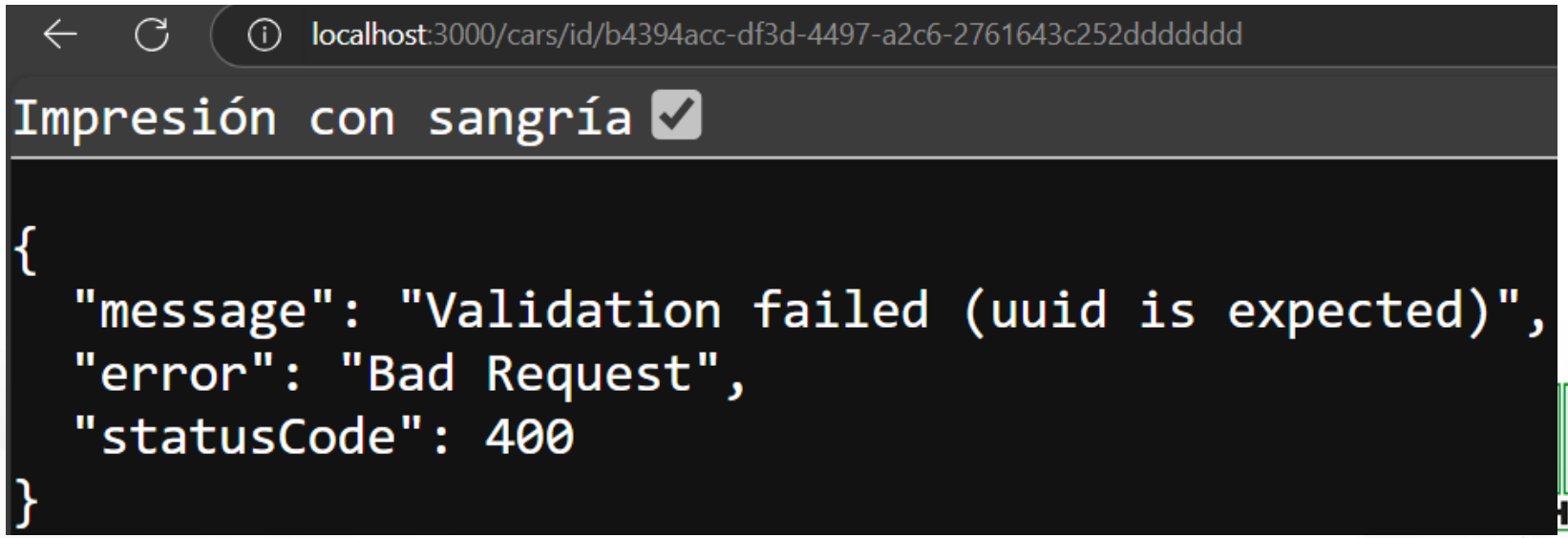


Fuente: Ing de Sistemas - Propia

Paso 4: Implementar ParseUUIDPipe

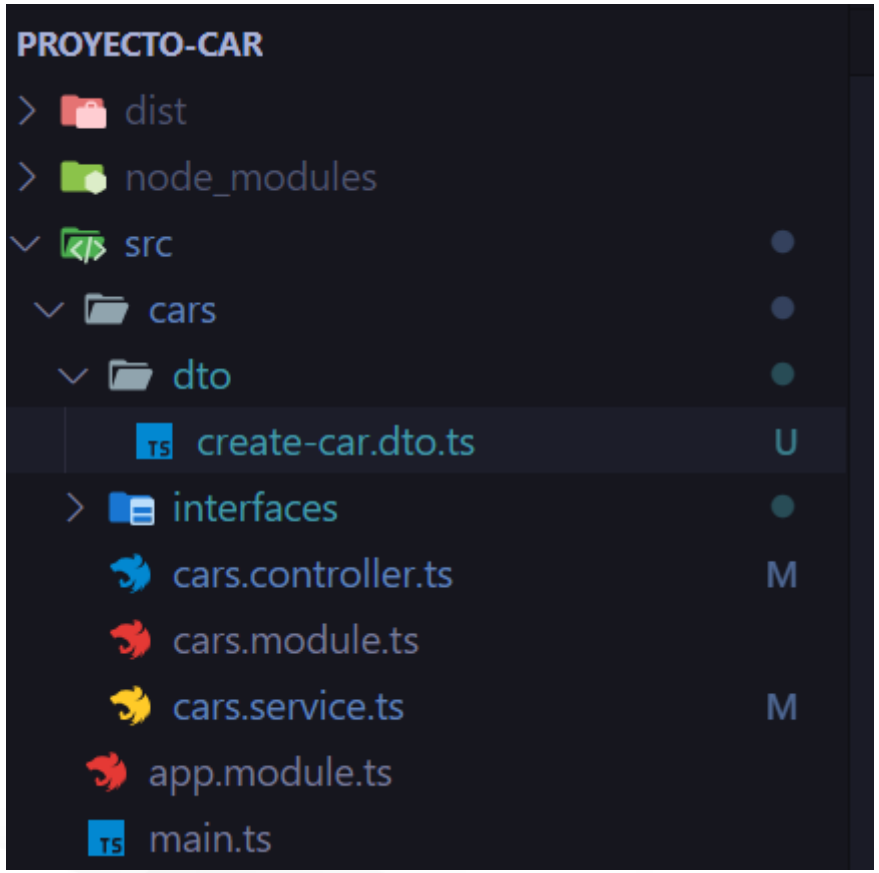
```
@Get('id/:id')
getCarById(@Param('id', ParseUUIDPipe)id:string){
    console.log({id})
    //throw new Error('Auxilio!')
    return this.carsService.findOneById(id);
}
```





Fuente: Ing de Sistemas - Propia

Paso 5: DTO (Data Transfer Object)




```typescript
export class CreateCarDto{
    readonly brand:string;

    readonly model:string;
}
```

```typescript
@Post()
createCar(@Body() createCarDto:CreateCarDto){
    return createCarDto;
}
```

Fuente: Ing de Sistemas - Propia

Paso 6: ValidationPipe

```
PS C:\Users\DELL\Desktop\ProyectoCar\proyecto-car> yarn add class-validator class-transformer
```

Algunos decoradores de Class Validator

IsOptional	IsPositive	IsMongoId
IsArray	IsString	IsUUID
IsDecimal	IsDate	IsDateString
IsBoolean	IsEmail	IsUrl

```
@Post()
@UsePipes(ValidationPipe)
createCar(@Body() createCarDto:CreateCarDto){
    return createCarDto;
}
```

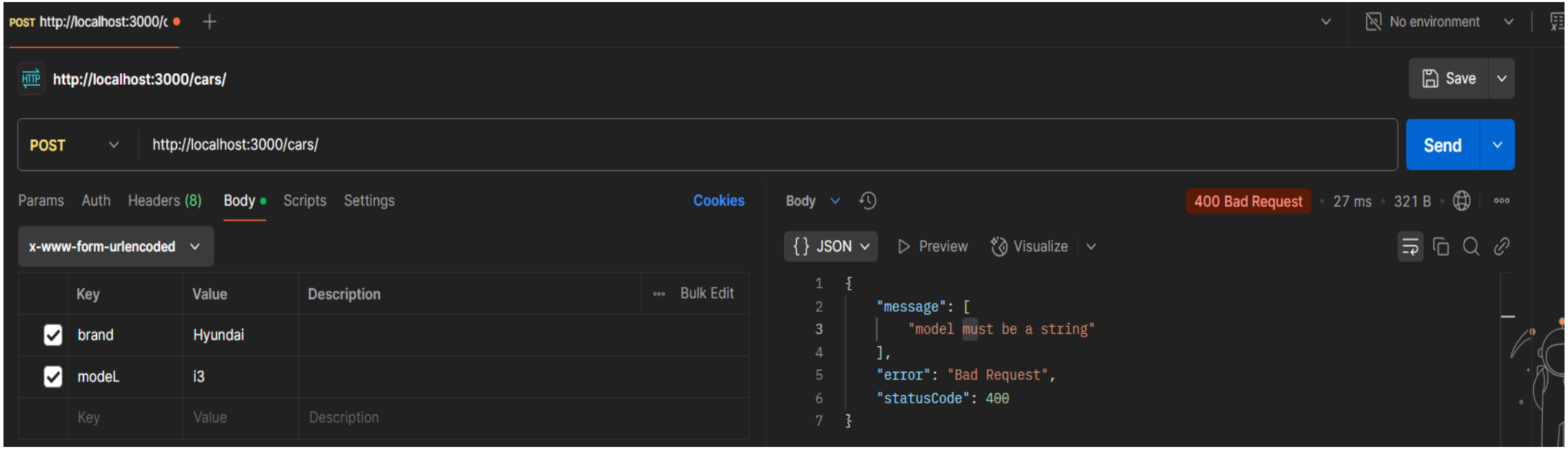
Pipes integrados por defecto

ValidationPipe	ParseIntPipe
ParseBoolPipe	ParseArrayPipe
ParseFloatPipe	ParseUUIDPipe

```
import { IsString } from "class-validator";

export class CreateCarDto{

    @IsString()
    readonly brand:string;

    @IsString()
    readonly model:string;
}
```

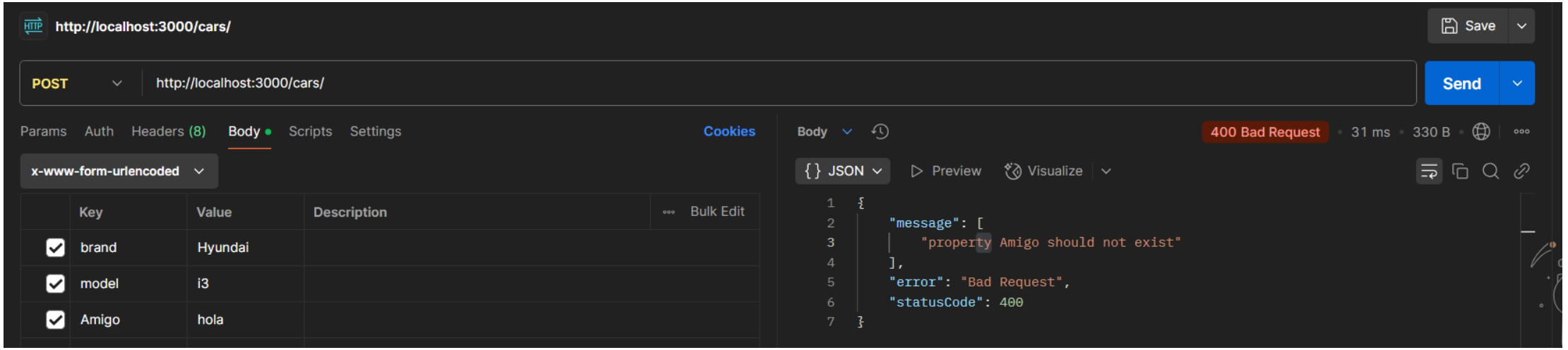
Fuente: Ing de Sistemas - Propia

Paso 6: ValidationPipe



Fuente: Ing de Sistemas - Propia

Paso 7: Pipes Globales



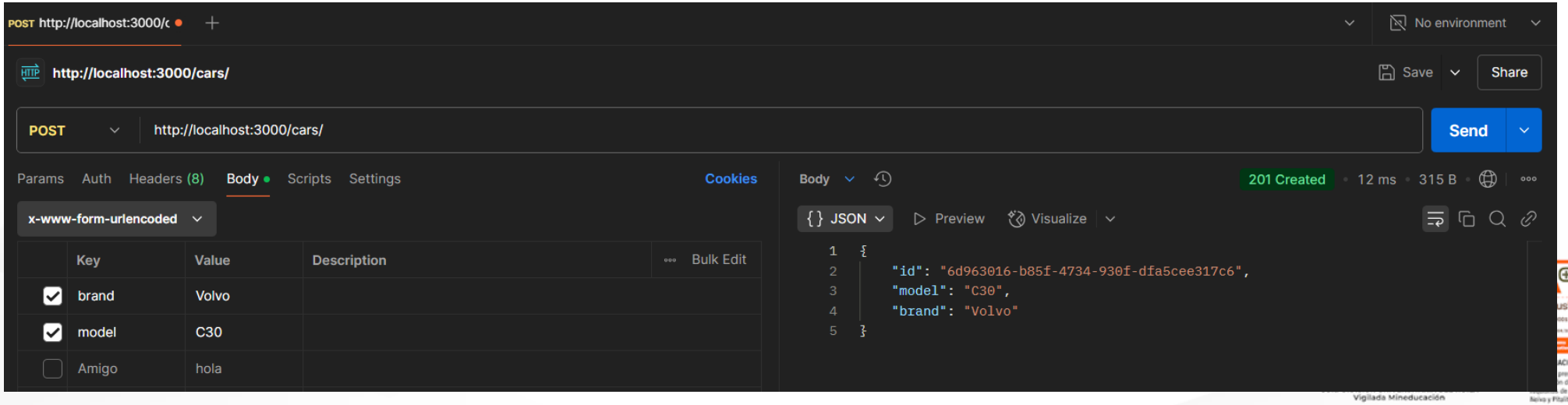

```
app.useGlobalPipes(
  new ValidationPipe({
  whitelist: true,
  forbidNonWhitelisted: true,
  })
  );
```

Fuente: Ing de Sistemas - Propia

Paso 8: Creando un Nuevo Carro

```
@Post()
//@UsePipes(ValidationPipe)
createCar(@Body() createCarDto:CreateCarDto){
    return this.carsService.create(createCarDto);
}
```

```
create(createCarDto: CreateCarDto){

    const car: Car={
        id: uuid(), //...createCarDto
        model:  createCarDto.model,
        brand: createCarDto.brand
    }

    this.cars.push(car);

    return car;
}
```



Fuente: Ing de Sistemas - Propia

CORHUILA

CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

GRACIAS

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA

06- Pipe Validation, Exception Filters, Post, Patch y Delete

Ing. Julián Quimbayo

Ingeniería de Sistemas

Facultad de Ingeniería

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

Paso 1: Montaje del Proyecto

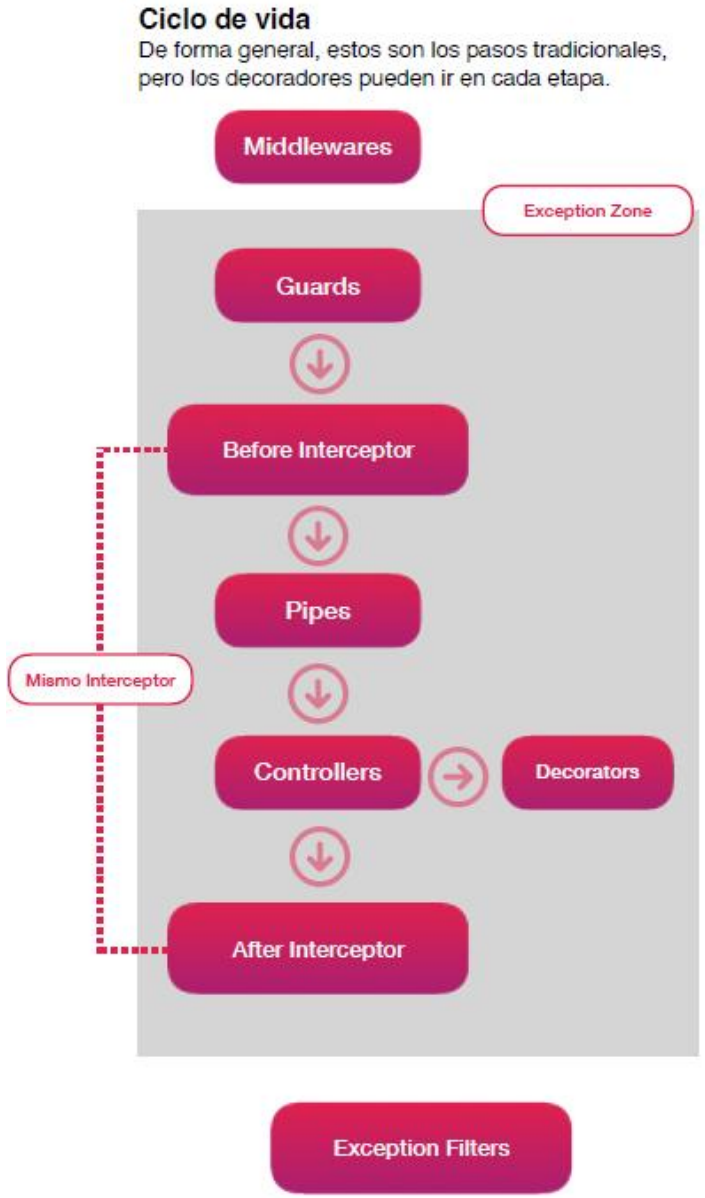
1. GitHub - jaquimbayoc7/ProyectoNest at master

Clonar el proyecto de GitHub

```
# Clonar el repositorio
git clone https://github.com/jaquimbayoc7/ProyectoNest.git

# Navegar al directorio del proyecto
cd ProyectoNest

# Instalar dependencias con npm
npm install

# O si prefieres usar yarn
yarn install
```

Paso 2: ¿Qué es un Pipe?

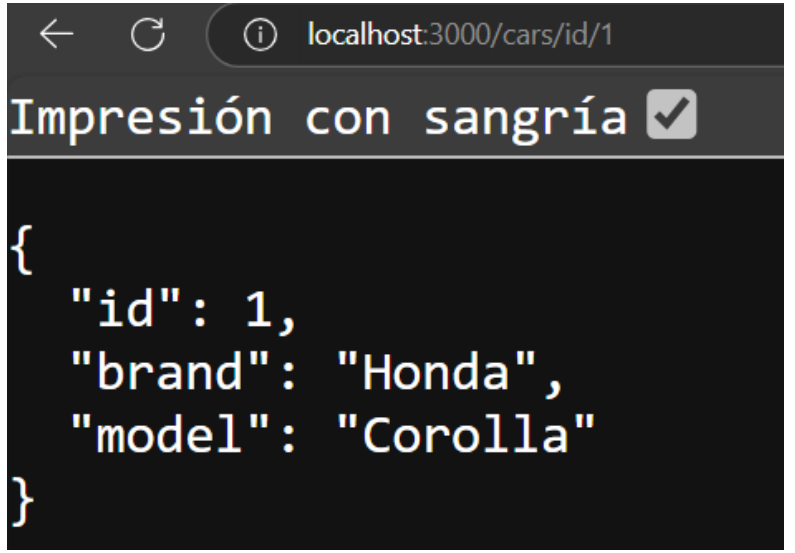
Pipes:
Transformar la data recibida en requests, para asegurar un tipo, valor o instancia de un objeto.
Ej: Transformar a números, validaciones, etc.

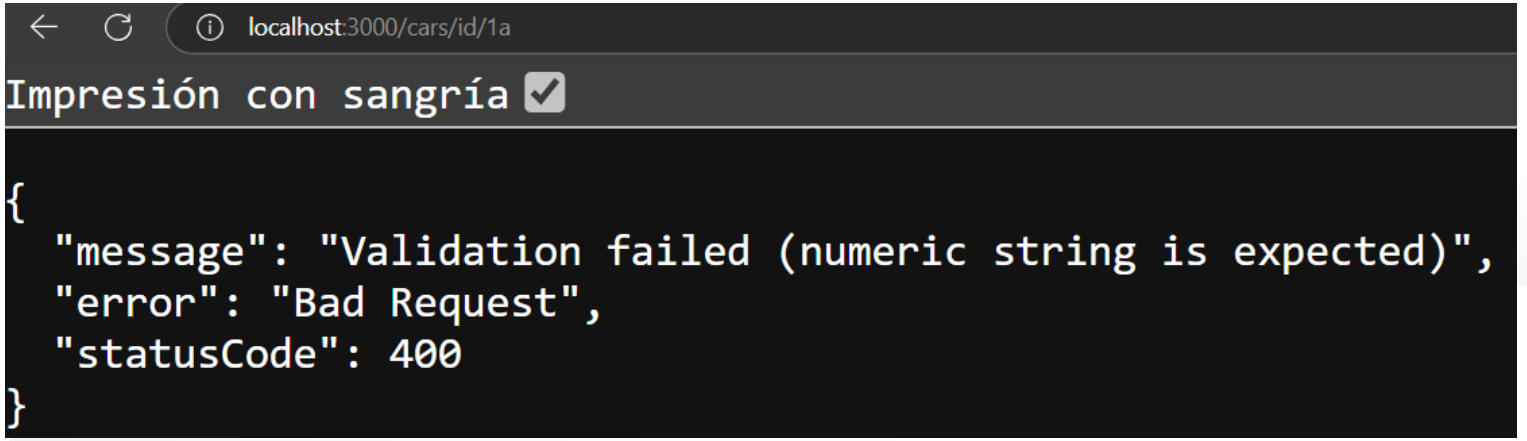


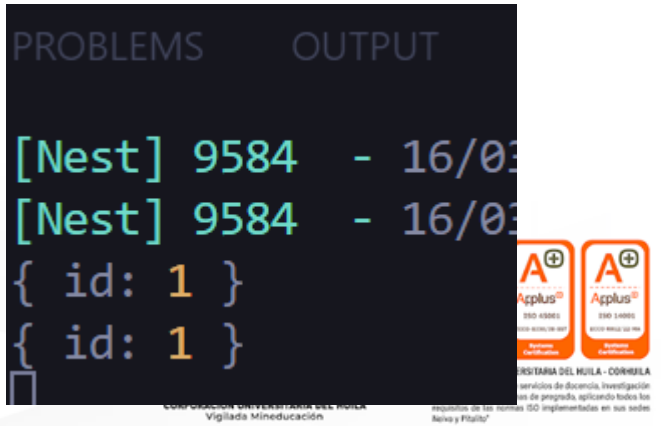
Fuente: https://www.youtube.com/watch?v=QNO6X4PCwTw

Paso 2: ¿Qué es un Pipe?

```
@Get('id/:id')
getCarById(@Param('id', ParseIntPipe)id){
    console.log({id})
   // throw new Error('Auxilio!')
    return this.carsService.findOneById(+id);
}
```







Fuente: Ing de Sistemas - Propia

Paso 2: ¿Qué es un Pipe?

```
@Get('id/:id')
getCarById(@Param('id', ParseIntPipe)id){
    console.log({id})
    throw new Error('Auxilio!')
    return this.carsService.findOneById(+id);
}
```

```
[Nest] 26452  - 16/03/2025, 12:06:14 p. m.   ERROR [ExceptionsHandler] Auxilio!
```

localhost:3000/cars/id/1

Impresión con sangría ✅


```
{
  "statusCode": 500,
  "message": "Internal server error"
}
```


Fuente: Ing de Sistemas - Propia

Paso 3: Exception Filters

- BadRequestException (400): Se lanza cuando la solicitud del cliente tiene errores de sintaxis o no puede ser procesada. Por ejemplo, cuando faltan campos requeridos o tienen un formato incorrecto.
- UnauthorizedException (401): Indica que el cliente debe autenticarse para obtener acceso. La solicitud carece de credenciales válidas o no se han proporcionado.
- NotFoundException (404): Se produce cuando el recurso solicitado no existe. Por ejemplo, cuando se busca un usuario o producto con un ID que no está en la base de datos.
- ForbiddenException (403): El servidor entiende la autenticado, pero se niega a autorizarla. El cliente está autenticado pero no tiene permisos suficientes.
- RequestTimeoutException (408): El servidor agotó el tiempo de espera mientras esperaba la solicitud del cliente.
- GoneException (410): Similar a 404, pero indica que el recurso solicitado existía anteriormente pero ya no está disponible y no volverá a estarlo.
- PayloadTooLargeException (413): La solicitud es más grande de lo que el servidor está dispuesto o puede procesar. Por ejemplo, al intentar subir un archivo demasiado grande.
- InternalServerErrorException (500): Error genérico del servidor que ocurre cuando se produce una condición inesperada que impide completar la

Exception Filters:

Maneja los errores de código en mensajes de respuesta http. Usualmente Nest ya incluye todos los casos de uso comunes, pero se pueden expandir basado en las necesidades.

BadRequestException	UnauthorizedException
NotFoundException	ForbiddenException
RequestTimeoutException	GoneException
PayloadTooLargeException	InternalServerErrorException

Fuente: https://www.youtube.com/watch?v=QNO6X4PCwTw

Paso 3: Exception Filters

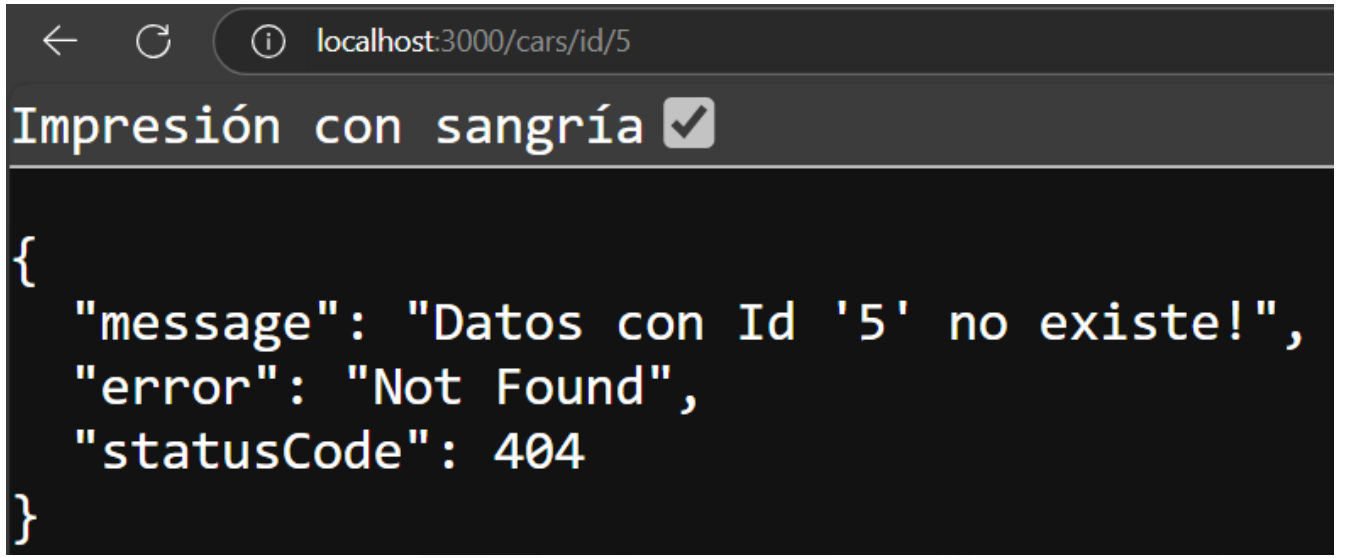
```
findOneById(id:number){
    const car = this.cars.find(car => car.id===id);

    if(!car){
        throw new NotFoundException();
    }

    return car;
}
```

Fuente: Ing de Sistemas - Propia

Paso 3: Exception Filters

```
findOneById(id:number){
    const car = this.cars.find(car => car.id===id);

    if(!car){
        throw new NotFoundException(`Datos con Id '${id}' no existe!`);
    }

    return car;
}
```

Fuente: Ing de Sistemas - Propia



Paso 4: Post, Patch y Delete

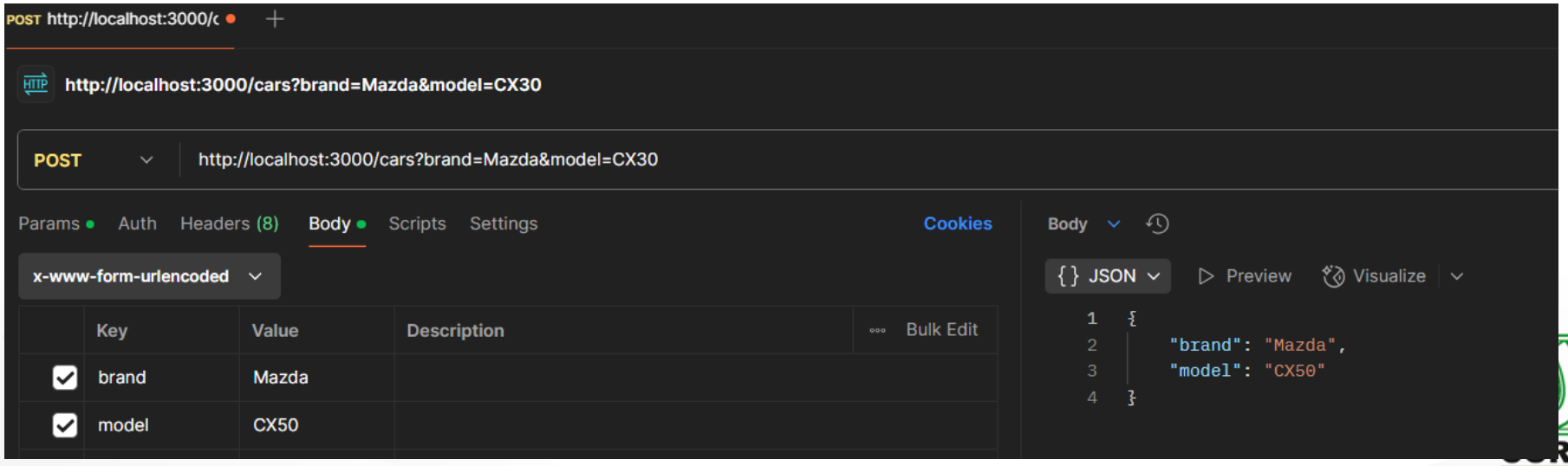
```
@Post()
createCar(@Body() body:any){
    return body;
}
```

Formulario

Brand: Mazda

Model: cx50

Guardar



Fuente: Ing de Sistemas - Propia

Paso 4: Post, Patch y Delete

```
@Patch('id/:id')
updateCar(
    @Param('id', ParseIntPipe) id:number,
    @Body() body:any){
    return body;
}
```

Oculto

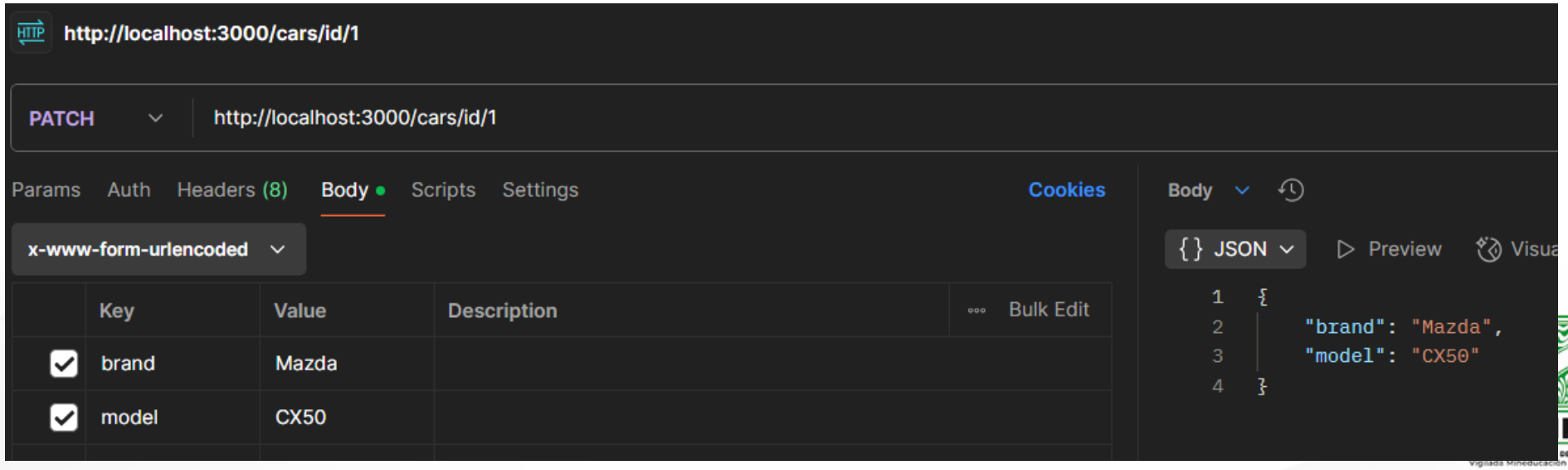
Id = 1

Formulario

Brand: Mazda

Model: cx50

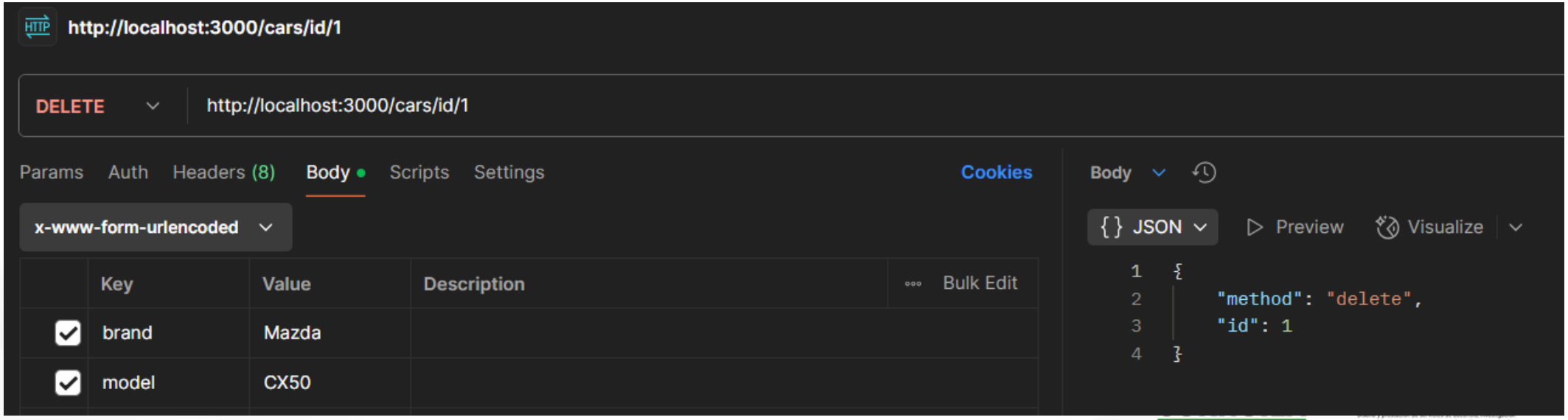
Actualizar



Fuente: Ing de Sistemas - Propia

Paso 4: Post, Patch y Delete

```
@Delete(':id')
deleteCar(@Param('id', ParseIntPipe) id:number){
    return{
        method:'delete',
        id
    }
}
```



Fuente: Ing de Sistemas - Propia

CORHUILA

CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

GRACIAS

CORHUILA
CORPORACIÓN UNIVERSITARIA DEL HUILA
Vigilada Mineducación

CORPORACIÓN UNIVERSITARIA DEL HUILA - CORHUILA