# MODELADO Y ARQUITECTURA DEL BACKEND

## Prework

### Introduccion

#### Objetivos

##### Comprender la importancia de la planificación y diseño de un proyecto.

##### Planificar un proyecto.

##### Diseñar un proyecto.

##### Reconocer los requerimientos de un proyecto.

##### Identificar las herramientas que se utilizarán para el desarrollo del proyecto.

##### Conocer la arquitectura usada en internet para el intercambio de información.

### Requerimientos

#### ¿Qué es un requerimiento?

##### un requerimiento expresa una necesidad y restricción que debe tener un producto de software para contribuir a resolver un problema del mundo real. Los requerimientos de un producto de software son las descripciones de lo que hará, los servicios que proporcionará y las restricciones de su operación.

##### Un requerimiento o necesidad es lo que el cliente o un usuario necesita que haga el software para resolver un problema.

#### Diseño de software

##### El diseño de cualquier producto consiste en crear un modelo o representación de lo que se construirá más adelante.

##### Diseño de software es el conjunto de actividades creativas mediante las cuales los requerimientos se traducen en una representación del software.

#### Objetivos del diseño

##### dentificar y caracterizar las partes o componentes principales del software.

##### Definir su interacción e integración en el producto.

##### Llegar al nivel de detalle que permite su mapeo al código de un lenguaje de programación.

#### Niveles de abstracción

##### Diseño arquitectónico que describe cómo se descompone y organiza el software en componentes abstractos partiendo de la especificación de requerimientos. Diseño detallado que describe el comportamiento específico de esos componentes tomando en cuenta el ambiente en el cual se codificará.

### Arquitectura de un proyecto

#### El ciclo de desarrollo de la arquitectura

##### “La arquitectura de software de un sistema es el conjunto de estructuras necesarias para razonar sobre el sistema. Comprende elementos de software, relaciones entre ellos, y propiedades de ambos.” (Bass, Clements y Kazman, 2012).

##### Dentro de un proyecto de desarrollo, e independientemente de la metodología que se utilice, se puede hablar de desarrollo de la arquitectura de software. Este desarrollo, que precede a la construcción del sistema, está dividido en las siguientes etapas: requerimientos, diseño, documentación y evaluación.

##### Cabe señalar que las actividades relacionadas con el desarrollo de la arquitectura de software generalmente forman parte de las actividades definidas dentro de las metodologías de desarrollo.

##### Aunque seamos los mejores programadores, al iniciar a construir una aplicación de software podemos enfrentarnos al problema de no saber por dónde comenzar si no tenemos los fundamentos necesarios de arquitectura de software. Por esto es necesario conocer el ciclo de la arquitectura de software y lo que involucra. Este ciclo está compuesto por las siguientes etapas:

###### Requerimientos

La etapa de requerimientos se enfoca en la captura, documentación y priorización de requerimientos que influencian la arquitectura. Como se mencionó anteriormente, los atributos de calidad juegan un papel preponderante dentro de estos requerimientos, así que esta etapa hace énfasis en ellos. Otros requerimientos, sin embargo, son también relevantes para la arquitectura, estos son los requerimientos funcionales primarios y las restricciones.

###### Diseño

La etapa de diseño es la etapa central en relación con la arquitectura y probablemente la más compleja. Durante esta etapa se definen las estructuras que componen la arquitectura. La creación de estas estructuras se hace en base a patrones de diseño, técnicas de diseño y elecciones tecnológicas. El diseño que se realiza debe buscar ante todo satisfacer los requerimientos que influencian a la arquitectura, y no simplemente incorporar diversas tecnologías porque están “de moda”.

###### Documentación

Una vez creado el diseño de la arquitectura, es necesario poder comunicarlo a otros involucrados dentro del desarrollo. La comunicación exitosa del diseño muchas veces depende de que dicho diseño sea documentado de forma apropiada. La documentación de una arquitectura involucra la representación de varias de sus estructuras que son representadas a través de distintas vistas. Una vista generalmente contiene un diagrama, además de información adicional, que apoya en la comprensión de dicho diagrama.

### Servidores

#### ¿Qué es un servidor?

##### El término servidor tiene dos significados en el ámbito informático.

###### El primero hace referencia al ordenador que pone recursos a disposición a través de una red,

###### el segundo se refiere al programa que funciona en dicho ordenador. En consecuencia aparecen dos definiciones de servidor:

- Definición de servidor (hardware)

Un servidor basado en hardware es una máquina física integrada en una red informática en la que, además del sistema operativo, funcionan uno o varios servidores basados en software. Una denominación alternativa para un servidor basado en hardware es host (término inglés para "anfitrión"). En principio, todo ordenador puede usarse como "host" con el correspondiente software para servidores.

- Definición de servidor (software)

Un servidor basado en software es un programa que ofrece un servicio especial que otros programas denominados clientes (clients) pueden usar a nivel local o a través de una red. El tipo de servicio depende del tipo de software del servidor. La base de la comunicación es el modelo cliente-servidor y, en lo que concierne al intercambio de datos, entran en acción los protocolos de transmisión específicos del servicio.

#### ¿Cómo funciona un servidor?

##### La puesta a disposición de los servicios del servidor a través de una red informática se basa en el modelo cliente-servidor, concepto que hace posible distribuir las tareas entre los diferentes ordenadores y hacerlas accesibles para más de un usuario final de manera independiente. Cada servicio disponible a través de una red será ofrecido por un servidor (software) que está permanentemente en espera. Este es el único modo de asegurar que los clientes, como el navegador web o los clientes de correo electrónico, siempre tengan la posibilidad de acceder al servidor activamente y de usar el servicio en función de sus necesidades.

##### La comunicación entre un servidor y sus clientes se basa en HTTP, es decir, en el protocolo de transferencia de hipertexto o en su variante codificada HTTPS. Además, ten en cuenta que toda computadora, smartphone o tablet tiene una dirección IP única e irrepetible que lo identifica de otro dispositivo en la red, así es como el servidor web envía la información exacta que el internauta está esperando.

##### Ahora bien, para que el web server pueda cumplir con su función es necesario que reciba la petición por parte de un navegador, en otras palabras, se envía un pedido desde una dirección IP hacia la dirección IP del servidor que aloja los archivos del sitio en cuestión.

##### A continuación, el servidor web busca en sus archivos la información que se le está solicitando, procede a interpretar las líneas de código y a enviar el resultado al navegador cuya dirección IP fue la solicitante.

##### Este resultado se le muestra a los internautas y es lo que siempre sucede cuando se navega en sitios de Internet. Cuando este proceso se completa podemos decir que el web server ha cumplido con su función.

#### ¿Para qué sirve un servidor en Internet?

##### El rol principal de un servidor web es almacenar y transmitir el contenido solicitado de un sitio web al navegador del usuario.

##### Este proceso, para los internautas no dura más que un segundo, sin embargo, a nivel del web server es una secuencia más complicada de lo que parece. Para cumplir con sus funciones el servidor deberá tener la capacidad de estar siempre encendido para evitar interrumpir el servicio que le ofrece a sus clientes. Si dicho servidor falla o se apaga, los internautas tendrán problemas al ingresar al sitio web.

### Tipos de Servidores

#### ¿Qué tipos de servidores hay?

##### Los servidores Web, también conocidos como servidores HTTP, surgen en 1990 ante la necesidad de estandarizar la comunicación entre distintas plataformas y lenguajes de programación, brindando una alternativa para el intercambio de información más rápida y sencilla.

##### Éstos proporcionan las herramientas necesarias para almacenar toda la información contenida en una página de internet, y de igual modo enviar la información al cliente que la solicite, es decir, hacen posible lo que se conoce como hosting.

##### Un servidor Web tiene la capacidad de atender simultáneamente las solicitudes de varios ordenadores. Sin embargo, la eficiencia de éste puede depender de factores como la velocidad de procesamiento, el hardware, el número de solicitudes y los contenidos web dinámicos, ya que estos últimos requieren una mayor cantidad de recursos que los contenidos estáticos.

##### Dentro de las opciones de servidores web que podrás encontrar, los más comunes son los servidores compartidos, los dedicados, y los servidores virtuales.

###### Los servidores compartidos ofrecen un espacio para que puedas alojar tu información, y debes compartir recursos como el mismo CPU, la memoria RAM, el sistema operativo, la dirección IP, e incluso la misma conexión a internet.

###### Un servidor dedicado es aquel que contrata un solo cliente, el cual no debe compartir recursos con nadie más, y la capacidad de almacenamiento que te ofrece es superior a cualquier otro tipo de alojamiento web.

###### Finalmente, el servidor virtual es el que combina características de los dos anteriores; el servidor físico realiza particiones virtuales que trabajan de forma independiente, y lo único que se comparte es el costo.

#### Servidores FTP

##### Un servidor de Protocolo para Transferencia de Archivos, o File Transfer Protocol por sus siglas en inglés, es un programa que se ejecuta cuando se está conectado a Internet o a algún otro tipo de red (LAN, MAN, etc.), el cual permite el desplazamiento o transferencia de datos entre otros servidores u ordenadores.

##### Este tipo de servidores se crearon tomando como base una arquitectura tipo cliente-servidor, es decir, la computadora del cliente debe conectarse primero con el servidor para poder descargar o almacenar archivos.

##### A pesar de que es uno de los protocolos más antiguos, ya que su creación data de 1971, sigue siendo uno de los más utilizados debido a que ha ido evolucionando con el paso del tiempo y es considerado rápido, confiable y seguro.

##### Los Servidores FTP son usados principalmente para el respaldo de datos y archivos, es decir, como un servidor de backup; sin embargo cuenta con otros usos.

#### Servidores de Correo

##### Los servidores de correo, son aplicaciones informáticas que permiten el intercambio de mensajes a través de sistemas de comunicación electrónicos, los cuales se comenzaron a desarrollar desde 1971.

##### Éstos trabajan como el servicio de correos, la gran diferencia es que en lugar de tardar días o semanas en enviar grandes paquetes y documentos en papel, envían datos digitales e información electrónica en cuestión de segundos, permitiendo una comunicación casi inmediata.

##### Para que se pueda llevar a cabo el correcto funcionamiento de estos servicios es necesario hacer uso de diferentes tipos de protocolos de red, que son básicamente las normativas o reglas que permiten que circule información en una red informática. Los más comunes son los protocolos HTTP, SMTP, POP, POP3, e IMAP.

#### Servidores IRC

##### Tuvo sus inicios en 1988 y fue un gran éxito debido a que logró generar una comunicación escrita inmediata y en tiempo real entre dos o más personas a través de internet.

##### El Internet Relay Chat por sus siglas en inglés, es actualmente uno de los servicios de chat más utilizados, el cual consiste básicamente en que varios servidores están conectados a una red, de modo que cualquier persona alrededor del mundo puede unirse a uno de éstos y chatear con cualquier otro usuario conectado a Internet. Al igual que un servidor FTP, éste también está basado en una arquitectura tipo cliente-servidor.

##### El IRC permite participar en salones de chat públicos y a la vez mantener conversaciones privadas; limitar el máximo de participantes e incluso expulsar a aquellos integrantes no deseados. Además, permite el intercambio de otro tipo de archivos, tales como imágenes o sonido, de modo que genera una mayor interacción en los salones de chat.

#### Servidores de Juegos

##### Los también conocidos Game Servers, como lo dice su nombre, se especializan en alojar juegos de video, es decir, proveer el espacio para que sus creadores o desarrolladores puedan ofrecer sus juegos en internet. Además, este servidor tiene como función recopilar la información de cada jugador conectado y enviarla al resto de participantes de forma inmediata de modo que puedan generarse partidas en tiempo real.

##### Existen dos tipos de servidores de videojuegos, el de escucha y el dedicado. Los servidores de escucha son utilizados en su mayoría por grupos reducidos de jugadores que desean tener sesiones de videojuegos compartiendo una red local. Funcionan ya que los requisitos de procesamiento y ancho de banda se ejecutan desde una misma máquina.

##### Los servidores dedicados trabajan de forma independiente, operados por el cliente desarrollador del videojuego. Éstos te permiten controlar y actualizar el contenido de forma constante, y su mayor ventaja es que, al estar ubicados en centros de datos, la potencia y el ancho de banda que ofrecen permiten que exista un gran número de jugadores simultáneos.

#### Servidores Proxy

##### Un servidor Proxy funciona como intermediario entre otros dos sistemas informáticos, contribuyendo a la seguridad de la red. Al no requerir comunicación directa entre el remitente y el destinatario, hace posible el intercambio de datos e información entre sistemas que cuentan con incompatibilidad de direcciones IP. Además, forma parte fundamental del Firewall, que es el software que protege a los sistemas de ataques de redes públicas como el robo de datos confidenciales, la pérdida de información valiosa, e incluso la denegación a servicios de nuestra red.

##### Entre otra de las funciones del proxy se encuentra la generación de lo que se conoce como Caché, el cual sirve para responder rápidamente las peticiones de una red local, y esto se logra mediante una copia temporal de los datos obtenidos desde Internet. Ayuda al ahorro de tiempo y ancho de banda.

#### VPN

##### Una Virtual Private Network es una tecnología utilizada para realizar conexiones seguras a una red privada de una o varias computadoras sin que se encuentren físicamente en el mismo lugar. Ésta ayuda a proteger nuestra información, ya que al conectarnos a una VPN se genera un tipo de “túnel virtual” por el que pasa toda nuestra información, y los datos enviados o solicitados se encuentran cifrados hasta que salen de la VPN.

##### Hacer uso de esta tecnología tiene ciertas ventajas. La primera y más importante de todas es, como ya se había mencionado anteriormente, la protección de tus datos. Sin embargo, otro de los grandes beneficios es el acceso a cualquier lugar de la red sin restricciones geográficas.

### Actividades

#### Actividad 01

##### Actividad 01. Gestores de versiones

##### Actividad 02. GitHub – Git hub Cheat Sheet

###### <https://training.github.com/downloads/es_ES/github-git-cheat-sheet.pdf>

##### Actividad 03. Frontend vs Backend

### Recursos

#### [What Is Software Architecture? .](https://www.castsoftware.com/glossary/what-is-software-architecture-tools-design-definition-explanation-best#:~:text=Software%20architecture%20is%2C%20simply%2C%20the,the%20software%20into%20the%20future.)

##### Software architecture is, simply, the organization of a system. This organization includes all components, how they interact with each other, the environment in which they operate, and the principles used to design the software. In many cases, it can also include the evolution of the software into the future.

##### Software architecture is designed with a specific mission or missions in mind. That mission has to be accomplished without hindering the missions of other tools or devices. The behavior and structure of the software impact significant decisions, so they need to be appropriately rendered and built for the best possible results.

#### ¿Qué es un servidor Web para qué sirve?. [Link](https://www.webempresa.com/hosting/que-es-servidor-web.html)

##### Un servidor web es un software que forma parte del servidor y tiene como misión principal devolver información (páginas) cuando recibe peticiones por parte de los usuarios.

##### En otras palabras, es el software que permite que los usuarios que quieren ver una página web en su navegador puedan hacerlo..

##### Funcionamiento de un servidor web?.

###### Para el funcionamiento correcto de un servidor web necesitamos un cliente web que realice una petición http o https a través de un navegador como Chrome, Firefox o Safari y un servidor donde esté almacenada la información.

###### El proceso sería el siguiente:

###### Tras la primera consulta por parte del usuario hacia una web, se establece una conexión entre el servidor DNS y el ordenador que realiza la consulta o petición. Este servidor DNS responde con la dirección IP correcta del servidor web donde está alojado el contenido solicitado.

##### .

## Work

### Sesión 01 - Modelado y arquitectura de un proyecto de Backend 🕹

#### Planificación de un proyecto.

##### En esta etapa se tiene que determinar la definición y alcance del proyecto y la organización del desarrollo del mismo, se dan fechas tentativas de entregas parciales. También es importante definir las herramientas que se utilizarán a lo largo del desarrollo del proyecto, tanto las herramientas técnicas (lenguajes de programación, bibliotecas, etc.) así como las de organización.

##### Ejemplo 01: Alcance y planificación del proyecto

###### Desarrollo

Antes de comenzar a definir el alcance que tendrá el proyecto es necesario entender que es lo que hace una aplicación de backend.

Hasta ahora en el curso se había estado trabajando unicamente con la vista de las aplicaciones web, es decir, con la interfaz a la que los usuarios y usuarias tienen acceso y mediante la cual interactua con nuestra aplicación. A este trabajo se le conoce como frontend.

El backend es la parte del desarrollo web que se encarga de que toda la lógica de una página web funcione. Se trata del conjunto de acciones que pasan en una web pero que no vemos. Es la parte que se conecta con la base de datos y el servidor que utiliza dicho sitio web, por eso decimos que el backend corre del lado del servidor.

Backend es la capa de acceso a los datos, que no es directamente accesible por los usuarios, además contiene la lógica de la aplicación que maneja dichos datos.

El Backend está compuesto por 3 partes fundamentales:

Servidor

Base de Datos

Middleware

###### Definición del alcance del proyecto.

Alcance

El alcance de un proyecto se define respondiendo una simple pregunta

El alcance de un proyecto se define respondiendo una simple pregunta

Esta pregunta es muy ambigua, y para ayudarnos a responderla primero definiremos cada uno de los siguientes puntos:

Objetivos del proyecto

Fases del proyecto

Programación temporal

Objetivos del proyecto

Los objetivos del proyecto se refieren específicamente a ¿Que debe hacer? y estos casi siempre son dados por el cliente al cuál vamos a vender la aplicación. En este caso nosotros definiremos los objetivos generales del proyecto.

La aplicación será un backend para un marketplace con los siguientes objetivos:

Que los usuarios puedan vender y comprar productos.

Que se puedan escribir reseñas sobre los productos.

Que un usuario pueda consultar el historial de los productos que compro.

Que se incluya la información relevante de los productos como lo es: precio, descripción, categoría, marca, etc.

Que un vendedor pueda dar de alta nuevos productos.

Que un vendedor pueda dar de baja productos ya no existentes.

Es importante notar como estos objetivos están definidos lejos de la especificación técnica, es decir, no hablan de la implementación de ellos. Solo son frases que describen lo que el cliente quiere.

Fases del proyecto

Las fases se refieren a esas etapas que constituyen la creación del proyecto. Es importante definir cuál será el resultado de cada una de las fases, es decir, que obtendremos cuando termine cierta fase. Nuestro proyecto se va a dividir en las siguientes fases:

Planificación: La fase actual en la que se define el alcance del proyecto y las herramientas para el desarrollo.

Diseño: Fase en la que se definen los requerimientos y las especificaciones técnicas.

Desarrollo: Esta es la etapa que más tiempo requiere y es en la que se escribe el código del programa.

Testing: Una fase de pruebas sobre la implementación obtenida en la fase anterior.

Fixes: Correcciones de errores encontrados en la fase de testing.

Deploy: Liberación del proyecto, en esta fase el proyecto se publica para su uso por usuarios reales.

Mantenimiento: Después de la liberación del proyecto se debe seguir al pendiente de aparición de errores (bugs) para corregirlos.

Programación temporal

La programación temporal se refiere a la calendarización de las tareas y fases definidas para el proyecto. Aquí se dan fechas de entregas parciales del desarrollo del proyecto. Estas fechas deben respetarse para que la programación funcione correctamente.

##### Reto 01 – Candelizacion

###### Completado

###### 

##### Ejemplo 2: Definición de las herramientas

###### El rectángulo que se encuentra en la parte superior izquierda, define la aplicación de backend.

###### Dentro del backend tenemos una base de datos de PostgreSQL.

###### Y una API desarrollada en JavaScript con npm y con github como gestor de versiones que se comunica con la base de datos con sequelize.

###### Todo el backend está hosteado en Heroku.

###### El backend se comunica con un cliente a través de Express.js en un navegador.

###### En la parte inferior podemos ver que el equipo de desarrollo utiliza Slack y Trello para la comunicación y organización respectivamente.

###### Todas estas son las herramientas que exploraremos durante el módulo, en donde cada una nos servira para resolver tareas en especifico.

###### En el diagrama se puede observar la estructura completa que tendrá nuestro proyecto. Estos diagramas son de gran utilidad ya que representan de forma intuitiva la arquitectura del proyecto que se está desarrollando.

###### 

#### Diseño de un proyecto

##### Después del proceso de planificación del proyecto viene la etapa de diseño, en esta fase la estructura completa del proyecto los módulos que se van a utilizar, que elementos van a ser parte del proyecto, como se van a definir esos elementos y como van a interactuar entre si. En esta etapa ya es necesario comenzar a pensar en la implementación del proyecto para poder definirlo.

##### Ejemplo 03: Requerimientos del proyecto

###### Los requerimientos nos dicen como desarrolladorxs que es lo que tenemos que tomar en cuenta a la hora de crear una aplicación o programa. Son las especificaciones que nos proporciona el cliente sobre que se espera del software que vamos a desarrollar.

###### Describen a muy alto nivel las especificaciones y funcionalidades del sistema.

###### Por ejemplo nosotros vamos a desarrollar un proyecto llamado BeduShop, que será una aplicación web que permita a la gente vender y comprar productos, a grandes rasgos los requerimientos son los siguientes:

Publicar productos en venta.

Visualizar los productos que están en disponibles.

Permitir a los usuarios escribir valoraciones sobre los productos que compraron.

Que los vendedores puedan dar de alta y baja productos.

###### Estos requerimientos están basados directamente en los objetos del proyecto descritos en el ejemplo 1, sin embargo en esta ocasión ya están escritos de una forma un poco mas cercana a la implementación.

###### Es importante que la definición de estos requerimientos esté libre de ambigüedad, de esa forma tanto el cliente como el equipo de desarrollo tendrán el mismo contexto sobre lo que se espera de la aplicación. Por lo que es necesario definir una metodología que nos permita describir estos requerimientos de tal forma que sean comprensibles para las personas involucradas y fácilmente traducibles a código.

###### Historias de usuario

Para describir los requerimientos de un sistema a menudo se usan historias de usuarix que son descripciones cortas y simples de una funcionalidad contada desde la perspectiva de la persona que la desea (usuarix o clientx del sistema).

Estas historias tienen una estructura simple

"Como [rol de usuario] quiero [objetivo] para [beneficio]"

Y son utilizadas por varias metodologías para poder comprender mejor los requerimientos y alinearlos con el negocio.

Ejemplo

Como usuaria de BeduShop, quiero ver a todos los productos en venta para poder elegir los que más me gusten.

Como vendedora de BeduShop, quiero agregar nuevos productos, para poder aumentar mi catalogo de opciones

Las historias de usuario están definiendo el comportamiento de lxs usuarixs o entidades dentro de la aplicación y nos permiten determinar cuales son las funciones de cada uno. Posteriormente estas historias se traducen a funcionalidades de nuestro sistema.

##### Reto 02: Historias de Usuario

###### .

##### Ejemplo 04: Modelo Vista Controlador

###### Desarrollo

Una de las maneras para bajar nuestras ideas y comenzar a diseñar la arquitectura de nuestra solución de software es por medio del patrón MVC (Model View Controller)

###### ¿Qué es MVC?

Es un patrón de diseño que separa las partes de nuestra aplicación en tres elementos.

Modelo: Contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.

Vista: Compone y presenta la información que se envía al cliente y los mecanismos interacción con éste por medio de una interfaz de usuario.

Controlador: Actúa como un intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

###### Flujo MVC

MVC es un estilo de arquitectura que nos sirve para abstraer el funcionamiento de nuestra aplicación y separar las partes referentes al negocio de la lógica. Actualmente existen varios frameworks que han adaptado este estilo a su manera y que nos ayudan a no perder tiempo y comenzar a desarrollar con reglas preestablecidas. Algunos de estos frameworks son:

SailJS o Express para NodeJS.

Django si lo tuyo es Python.

Ruby on Rails para el lenguaje de programación Ruby.

Laravel si lo tuyo es PHP.

Los cuales nos permiten entregarle al usuario las vistas (documentos HTML, CSS y Javascript) desde el servidor.

MVC también ha sido adaptado para utilizarse en frontend y en Android.

###### Modelo: Identificación de modelos

Continuando con BeduShop, podemos identificar cuatro entidades principales:

Producto: Se refiere al producto que registran los vendedores y los usuarios pueden comprar.

Usuario: hay dos tipos de usuarios de nuestra aplicación, el tipo comprador y el tipo vendedor

Venta: Una venta define el momento en el que un comprador ordena un producto.

Reseña: Una valoración escrita por un usuario sobre un producto.

#### ¿Cómo nos comunicamos en la internet?

##### Al estar desarrollando un proyecto de backend es importante comprender como sucede el intercambio de información en internet, pues en el desarrollo de nuestro proyecto estaremos principalmente enviando y recibiendo datos por este medio.

##### Ejemplo 05: Arquitectura Cliente- Servidor

###### Servidor web

Un servidor web es un conjunto de software y hardware que responden las peticiones que los clientes hacen sobre World Wide Web.

Para responder a las peticiones, los servidores utilizan distintos protocolos de transferencia de datos por una red, siendo los principales:

HTTP/HTTPS

SMTP

FTP

Su propósito principal es permitir el acceso al contenido de los sitios web que requieren lxs usuarixs, para esto, el servidor almacena, procesa y envía las paginas web.

###### ¿Cómo funciona un servidor web?

El hardware del servidor está conectado a la internet y permite el intercambio de información con otros clientes también conectados a la red. Este es una computadora que almacena el contenido del sitio web tales como los archivos HTML, JavaScript, CSS, imagenes, etc, es decir, es un host.

Mientras que el software controla el acceso que tienen lxs usuarixs a los archivos del servidor. Todo esto utilizando un modelo cliente servidor. Se accede a este mediante la url (Uniform Resource Locator) del sitio, que sirve como el localizador del sitio y asegura que el contenido será entregado a quienes que lo solicitaron.

Todo este proceso se hace utilizando el modelo cliente/servidor.

###### Proceso

Cuando entramos al navegador y colocamos la dirección de nuestro sitio web favorito, el navegador hace el siguiente proceso para encontrar la página que le pedimos:

Con la url, identifica la dirección IP del servidor en el que se hostea el sitio que le pedimos

Hace una solicitud de los archivos necesarios con el protocolo HTTP

El servidor acepta la petición, busca los archivos y los envía como respuesta

🔐 Arquitectura Cliente-servidor

La arquitectura que gobierna la web actualmente es la arquitectura cliente-servidor.

Cliente

El cliente se ocupa de hacer peticiones, recibir respuestas y presentarlas al usuario. En los primeros días eran ordenadores de uso común, ahora un cliente es cualquier tipo de dispositivo capaz de enviar una petición, esto engloba smartphones y dispositivos inteligentes como bocinas, luces, refrigeradores, relojes, termostatos, etc.

Servidor

Es un sistema diseñado específicamente para satisfacer las demandas de información de los clientes. El servidor recibe las peticiones del cliente, las procesa y responde la información solicitada.

Los servidores suelen realizar tareas complejas y especializadas, comúnmente también hacen peticiones a otros sistemas como a servidores de bases de datos o servicios externos e internos (micro-servicios).

###### ✉️ Protocolo HTTP

En computación, un protocolo es únicamente una manera en la que acordamos que se comunicará un sistema.

Haciendo una analogía podemos decir que un protocolo en la vida real sería la serie de reglas del sistema postal de correo. Si hoy quisiéramos enviar una carta por correo necesitaríamos escribir en un sobre el nombre y la dirección del destinatario, esta dirección a su vez contendría su código postal, también necesitaríamos un timbre y los datos del remitente.

De manera similar, el protocolo base para el funcionamiento de la web es el protocolo HTTP, que significa "Hypertext Transfer Protocol". Este protocolo de petición-respuesta está basado en otros protocolos que funcionan en un nivel más bajo de la red.

Peticiones

HTTP define un conjunto de métodos de petición (request method) para indicar que acción se desea realizar. Los métodos más importantes son:

Get - Solicita una representación de un recurso específico. Las peticiones que usan el método GET únicamente obtienen datos.

POST - El método POST se utiliza para enviar una entidad a un recurso en específico, causando a menudo un cambio en el estado o efectos secundarios en el servidor. También es común que se utilice para crear nuevos registros de recursos en una API.

PUT - El modo PUT reemplaza todas las representaciones actuales del recurso de destino con la carga útil de la petición.

DELETE - El método DELETE borra un recurso en específico.

Estos métodos proporcionan las llamadas operaciones CRUD o simplemente el CRUD y son:

Create

Read

Update

Delete

###### Respuestas

Las respuestas además de que pueden estar conformadas opcionalmente por un cuerpo o contenido, son definidas por un código de respuesta. Los códigos de respuesta indican si una petición se ha completado exitosamente o no, y nos brindan información sobre el estado de la respuesta. Las respuestas se dividen en 5 categorías:

Informativas (100-199) Usualmente se utilizan para informar que se recibió la petición o información .

Éxito (200–299) Indica que la petición fue recibida correctamente, entendida y aceptada.

Redirecciones (300–399) Le indican al cliente que es necesaria una acción de su parte para completar la petición.

Error del lado del cliente (400–499) La solicitud no puede procesarse por un error por parte del cliente, como puede ser un error en sintaxis o falta de algún header.

Error del lado del servidor (500–599) La solicitud era aparentemente válida, pero el servidor falló al completarla (errores de conexión, no se encontró la información, etc.).

Algunos de los códigos de respuesta más comunes son:

200 OK

Todo salio bien, es la respuesta estándar para peticiones correctas.

301 MOVED PERMANENTLY

El servidor se movió y ésta y todas las peticiones futuras deben ser dirigidas a la nueva URL.

302 FOUND

Se requiere que el cliente realice una redirección temporal.

404 NOT FOUND

El servidor web no puedo encontrar el recurso solicitado.

500 Internal Server Error

Ocurrió un error dentro del servidor al intentar resolver la petición. Es el código de error más común

.

##### Reto 03: Códigos HTTP

###### Código Descripción

###### 102 - El servidor recibio y esta procesando la solicitud pero todavia no termina

###### 202 - acceptado pero esta processando y todavia no termina

###### 206 - Esta entrango una parte del cliente

###### 300 - tenemos mas de una ultima respuestas

###### 304 - no ha sido actualizado

###### 400 - solicitud invalidad

###### 401 - la solicitud no fue verificada

###### 418 - Soy una tetera

###### 501 - Indica que el metodo que se uso a la solictud y no fue encontrada la solicitud

###### 511 - el cliente requiere acceso a network access.

## PostWork

### Desarrollo

#### Trello

##### Como se vió en el work, Trello es una herramienta muy útil para la organización y planificación de proyectos de software. Veamos qué otras posibilidades nos ofrece.

##### Empieza a colaborar

###### Para mejorar la colaboración a nivel de proyecto, Invita a miembros al tablero para asignarles tareas y que puedan colaborar en él. De esta forma tienes la certeza de que todos los miembros del equipo de desarrollo tienen la misma información además de que las actualizaciones se visualizan en tiempo real.

###### Para poder agregar a los miembros del equipo de desarrollo solo sigue los siguientes pasos.

#### Entregables

##### Para esta sesión vas a definir el proyecto personal en el que se estará trabajando todo el módulo. Este proyecto debe ser una aplicación de backend en donde el contenido del proyecto (los datos e información) son a elección del equipo.

##### Con este proyecto en mente, realizarás un documento en donde se incluyan los siguientes entregables:

###### Los objetivos generales del proyecto están pensados desde un punto de vista de un cliente que le solicita a un equipo de desarrollo la implementación del proyecto, es decir, objetivos no técnicos.

###### Definir en un tablero en Trello la planificación temporal del proyecto, basándose en la vista en el work. Pues el proceso de desarrollo será muy similar. Agregar un screenshot del tablero al documento. Utilizar también todo el conocimiento nuevo de trello de la sección anterior si es posible. Hay que considerar que la fecha de presentación del proyecto final será en la sesión de Checkpoint

###### Para la definición de las herramientas a utilizar, hacer una investigación de las herramientas propuestas en el work. Mencionar qué son y para qué sirven:

PostgreSQL

JavaScript

Node.js

npm

Sequelize

Express.js

Github

Heroku

##### Agregar un mínimo de 6 historias de usuario que definan las funcionalidades de estos dentro de la aplicación. Y explicar por qué son importantes estas funcionalidades.

##### Identificar correctamente los modelos del proyecto, cada uno con sus características específicas, dentro de una arquitectura modelo vista controlador

# INTRODUCCIÓN A BASES DE DATOS

## Pre-work

### Introducción

#### Setup

##### Cliente de PostgreSQL pgAdmin

##### Instalar docker en tu computadora siguiendo las instrucciones para la instalación de docker

### Bases de datos

#### ¿Qué es una base de datos?

##### Como se vió en la sesión anterior las bases de datos son un componente importante de una aplicación de backend. Pero para poder utilizarlas es necesario primero entender qué son y cómo funcionan.

#### Bases de datos relacionales

##### Una base de datos relacional es un tipo de base de datos que almacena y proporciona acceso a puntos de datos relacionados entre sí. Las bases de datos relacionales se basan en el modelo relacional, una forma intuitiva y directa de representar datos en tablas. En una base de datos relacional, cada fila de la tabla es un registro con un ID único llamado clave. Las columnas de la tabla contienen atributos de los datos, y cada registro generalmente tiene un valor para cada atributo, lo que facilita el establecimiento de las relaciones entre los puntos de datos.

#### ¿Cómo se estructuran las bases de datos relacionales?

##### El modelo relacional significa que las estructuras lógicas de datos, las tablas de datos, vistas e índices están separadas de las estructuras físicas de almacenamiento. Esta separación significa que los administradores de bases de datos pueden administrar el almacenamiento físico de datos sin afectar el acceso a esos datos como una estructura lógica.

##### Por ejemplo: cambiar el nombre de un archivo de base de datos no cambia el nombre de las tablas almacenadas en él.

##### La distinción entre lógica y física también se aplica a las operaciones de la base de datos, que son acciones claramente definidas que permiten a las aplicaciones manipular los datos y las estructuras de la base de datos. Las operaciones lógicas permiten que una aplicación especifique el contenido que necesita, mientras que las operaciones físicas determinan cómo se debe acceder a esos datos y luego realizan la tarea.

##### Para garantizar que los datos sean siempre precisos y accesibles, las bases de datos relacionales siguen ciertas reglas de integridad.

#### El modelo relacional

##### En los primeros años de las bases de datos, cada aplicación almacena datos en su propia estructura única. Cuando los desarrolladores querían crear aplicaciones para usar esos datos, tenían que saber mucho sobre la estructura de datos particular para encontrar los datos que necesitaban. Estas estructuras de datos eran ineficientes, difíciles de mantener y difíciles de optimizar para ofrecer un buen rendimiento de la aplicación. El modelo de base de datos relacional se diseñó para resolver el problema de varias estructuras de datos arbitrarias.

##### El modelo relacional proporciona una forma estándar de representar y consultar datos que cualquier aplicación podría utilizar. Desde el principio, los desarrolladores reconocieron que la principal fortaleza del modelo de base de datos relacional estaba en el uso de tablas, que eran una forma intuitiva, eficiente y flexible de almacenar y acceder a información estructurada.

##### Con el tiempo, cuando los desarrolladores comenzaron a utilizar el lenguaje de consulta estructurado (SQL) para escribir y consultar datos en una base de datos, surgió otra fortaleza del modelo relacional. Durante muchos años, se utilizó ampliamente el SQL como lenguaje para consultas de bases de datos. SQL, que se basa en el álgebra relacional, proporciona un lenguaje matemático internamente consistente que facilita la mejora del rendimiento de todas las consultas de la base de datos. En comparación, otros enfoques deben definir consultas individuales.

#### Beneficios de las bases de datos relacionales

##### Las organizaciones de todo tipo y tamaño utilizan el modelo relacional simple pero poderoso para una amplia variedad de necesidades de información. Las bases de datos relacionales se utilizan para hacer seguimiento de los inventarios, procesar transacciones de comercio electrónico, administrar grandes cantidades de información de clientes de misión crítica y mucho más. Se puede considerar una base de datos relacional para cualquier necesidad de información en la que los puntos de datos se relacionan entre sí y se deban administrar de una manera segura, consistente y basada en reglas.

##### Las bases de datos relacionales existen desde la década de 1970. Actualmente, las ventajas del modelo relacional continúan convirtiéndolo en el modelo más aceptado para bases de datos.

#### Consistencia de los datos

##### El modelo relacional es el mejor para mantener la consistencia de los datos en todas las aplicaciones y copias de la base de datos (denominadas instancias).

###### Por ejemplo:

Cuando un cliente deposita dinero en un cajero automático y, luego, mira el saldo de la cuenta en un teléfono móvil, el cliente espera ver que ese depósito se refleja inmediatamente en un saldo de cuenta actualizado. Las bases de datos relacionales se destacan en este tipo de consistencia de datos, lo que garantiza que múltiples instancias de una base de datos tengan los mismos datos todo el tiempo.

Es difícil para otros tipos de bases de datos mantener este nivel de coherencia oportuna con grandes cantidades de datos. Algunas bases de datos recientes, como NoSQL, solo pueden proveer consistencia eventual. Bajo este principio, cuando la base de datos se escala o cuando varios usuarios acceden a los mismos datos al mismo tiempo, los datos necesitan algo de tiempo para “ponerse al día.” La consistencia eventual es aceptable para algunos usos, como para mantener listados en un catálogo de productos, pero para operaciones comerciales críticas como transacciones de un carrito de compras, la base de datos relacional sigue siendo el estándar de oro.

#### Procedimientos almacenados y bases de datos relacionales

##### El acceso a los datos implica muchas acciones repetitivas. Por ejemplo, una consulta simple para obtener información de una tabla de datos puede necesitar repetirse cientos o miles de veces para producir el resultado deseado. Estas funciones de acceso a los datos requieren algún tipo de código para acceder a la base de datos. Los desarrolladores de aplicaciones no desean escribir un código nuevo para estas funciones en cada aplicación nueva. Afortunadamente, las bases de datos relacionales permiten procedimientos almacenados, que son bloques de código a los que se puede acceder con una simple llamada de aplicación. Por ejemplo, un solo procedimiento almacenado puede proporcionar un etiquetado de registro consistente para usuarios de varias aplicaciones. Los procedimientos almacenados también pueden ayudar a los desarrolladores a garantizar que ciertas funciones de datos en la aplicación se implementen de una manera específica.

#### Bloqueo de bases de datos y concurrencia

##### Pueden surgir conflictos en una base de datos cuando varios usuarios o aplicaciones intentan cambiar los mismos datos al mismo tiempo. Las técnicas de bloqueo y concurrencia reducen la posibilidad de conflictos mientras mantienen la integridad de los datos.

##### El bloqueo evita que otros usuarios y aplicaciones accedan a los datos mientras se actualizan. En algunas bases de datos, el bloqueo se aplica a toda la tabla, lo que crea un impacto negativo en el rendimiento de la aplicación. Otras bases de datos, como las bases de datos relacionales de Oracle, aplican bloqueos a nivel de registro, lo que deja disponibles los otros registros dentro de la tabla, lo que ayuda a garantizar un mejor rendimiento de la aplicación.

##### La concurrencia gestiona la actividad cuando varios usuarios o aplicaciones realizan consultas al mismo tiempo en la misma base de datos. Esta capacidad proporciona el acceso correcto a los usuarios y las aplicaciones de acuerdo con las políticas definidas para el control de datos.

#### Bases de datos no relacionales

##### NoSQL es un enfoque de diseño de bases de datos que nos permite crear esquemas más flexibles para el almacenamiento de datos en comparación con las tradicionales tablas de SQL. Las bases de datos no relacionales fueron presentadas en 1998 por Carl Strozz, sin embargo ha sido recientemente que han adquirido mayor popularidad y esto se debe principalmente a que vivimos en una era de Cloud, Big Data y web con información difícil de estandarizar que debe viajar a gran velocidad.

##### NoSQL significa Not Only SQL, no es nada en contra de SQL. Esto es importante pues muchos modelos de bases de datos no relacionales soportan SQL.

#### Tipos de DB NoSQL

##### De documentos Se almacenan los datos en estructuras llamadas documentos, estos contienen casa atributo como pares de clave-valor.

##### Los grafos se utilizan para almacenar información sobre redes de datos, como las conexiones sociales.

##### De clave-valor son las bases de datos NoSQL más simples. Cada elemento de la base de datos se almacena como un nombre de atributo junto con su valor.

##### Orientadas a columnas estas bases de datos permiten realizar consultas en grandes conjuntos de datos y almacenan los datos en columnas, en lugar de filas.

###### bases de datos 1

##### Bases de Datos NoSQL en el mercado

###### bases de datos 2

### SQL vs NoSQL

#### Diferencia Conceptual

##### Los sistemas de gestión de bases de datos relacional son sistemas multihilo y multiusuario con más de seis millones de instalaciones; usado por muchos sitios web grandes y populares, como Wikipedia, Google (no para búsquedas), Twitter, Flickr, y YouTube.

##### La diferencia conceptual entre SQL y NoSQL, es que resuelven escenarios completamente diferentes y excluyentes el uno del otro; ya que para lo que resulta ideal SQL, no lo es NoSQL y al revés.

###### sql_nosql

#### Diferencias

##### SQL permite combinar de forma eficiente diferentes tablas para extraer informacion relacionanda, mientras que NO SQL no lo permite o muy limitadamente

##### NOSQL permite distribuid grantes cantidates de información mientras que SQL facilita distribui bases de datos relacionales. Sql permite gestionar los datos junto con relaciones existentes entre ellos, en NoSQL no existe este tipo de utilidades

##### NoSQL permite un escalado horizontal sin problemas por su capacidad de distribución, mientas que escalar con SQL resulta mas complicado

#### ¿Cuándo usar SQL?

##### No existe una respuesta concreta, porque se presume que cualquier cosa que se deba guardar, debe ser en una base de datos relacional . Sin embargo los programadores podrían sugerir que en gaming o desarrollos de animación, no sería necesario.

##### En la mayoría de las opiniones, una base de datos relacional puede ser usada los siguientes ámbitos:

###### Educación: Para estructurar información y aportar conocimiento lógico al estudiante.

###### Negocios: Inteligencia y análisis de negocios, son temas que requieren el uso de SQL para facilitar el consumo de la información y la identificación de patrones en los datos.

###### Empresarial: Tanto el software a la medida y el software empresarial, poseen la característica de mantener información con estructura consistente .

##### Desventajas

###### Menos flexibilidad: Hay que planificar con tiempo el esquema ya que luego es más complicado actualizarlo.

###### El uso de relaciones: Puede conllevar consultas JOIN complejas. A veces el escalado horizontal se torna demasiado complicado.

##### ¿Cuándo usar NoSQL?

###### Redes sociales: Casi obligatorio.

###### Desarrollo Web: Debido a la poca uniformidad de la información que se encuentra en Internet; aun cuando también puede emplearse SQL.

###### Desarrollo Móvil: Debido a la tendencia en crecimiento de Bring Your Own Device.

###### BigData: Debido a la administración de grandísimas cantidades de información y su evidente heterogeneidad.

###### Cloud (XaaS): “Everything as a service”; NoSQL puede adaptarse casi a cualquier necesidad del cliente, y sus particularidades.

##### Desventajas

###### La flexibilidad puede conllevar a posponer decisiones de estructura importantes. Alta posibilidad de tener datos duplicados.

### Actividad 1

#### Familiarizándonos con SQL en Replit

##### SELECT 'Hello, Bedu!';

###### 

### Actividad 2

#### Create table – Creamos la tabla, no nos muestra ningún contenido

##### CREATE TABLE mountains (

##### name TEXT,

##### height\_meters INTEGER,

##### first\_ascent DATE

##### );

###### The CREATE TABLE statement is used to create a new table in a database.

###### The column parameters specify the names of the columns of the table.

###### The datatype parameter specifies the type of data the column can hold (e.g. varchar, integer, date, etc.).

###### Syntax

CREATE TABLE table\_name (

column1 datatype,

column2 datatype,

column3 datatype,

....

);

#### INSERT INTO – Insertamos valores y datos a nuestra tabla

##### Syntax 01 - 1. Specify both the column names and the values to be inserted:

###### INSERT INTO table\_name (column1, column2, column3, ...)

###### VALUES (value1, value2, value3, ...);

##### Syntax 02 - If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows

###### INSERT INTO table\_name

###### VALUES (value1, value2, value3, ...);

##### Usage

###### INSERT INTO mountains VALUES

###### ('Mount Everest', 8848, '1953-05-29'),

###### ('Kilimanjaro', 5895, '1889-10-06'),

###### ('Denali', 6190, '1913-06-07'),

###### ('Chimborazo', 6263, '1880-01-04'),

###### ('K2', 8611, '1954-07-31'),

###### ('Piz Palü', 3900, '1835-08-12'),

###### ('Cho Oyu', 8188, '1954-10-19');.

#### SELECT – se usa para selectionar información de la tabla

##### The SELECT statement is used to select data from a database.

##### The data returned is stored in a result table, called the result-set.

##### Syntax

###### SELECT column1, column2, ...

###### FROM table\_name;

Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

SELECT \* FROM table\_name;

### Recursos

#### <https://pandorafms.com/blog/es/nosql-vs-sql-diferencias-y-cuando-elegir-cada-una/>

##### ¿Qué es NoSQL?

###### Antes de analizar las diferencias concretas, de hablar de bases de datos relacionales y no relacionales, necesitamos asentar una definición clave que, en ocasiones, no termina de quedar clara.

###### ¿Sabes lo que significa NoSQL?

###### Lejos de ser algo complicado, lo que quiere decir es Not Only SQL (no solo SQL). Simplemente por aclararlo.

##### Diferencia entre NoSQL y SQL

###### La diferencia fundamental entre ambos tipos de bases de datos se explica de forma tan simple como que:

SQL implica bases de datos relacionales

No SQL implica bases de datos NO relacionales

###### Es decir, es el modelo de base de datos relacional o no relacional el que caracteriza a unas y a otras.

##### Base de datos relacional vs no relacional

###### Aclarados los conceptos esenciales, acompáñame a ver cuáles son las ventajas y desventajas de cada uno de los modelos de base de datos.

###### Esto te ayudará a tener una visión global de sus utilidades y, por tanto, te ayudará a tomar una decisión sobre cuál utilizar en tu caso.

##### 1. Bases de datos relacionales (SQL)

###### Dicho de manera sencilla, una base de datos relacional se basa en el almacenamiento y acceso a datos que están relacionados entre sí.

###### Es decir, tienen una forma muy directa e intuitiva de mostrarnos los datos que almacenan en forma de tablas.

###### Dentro de esas tablas, cada fila será un registro que tendrá un identificador único (conocido como clave) y las columnas incluirán atributos de esas claves. Normalmente esos atributos serán valores únicos, lo que nos permite crear esas relaciones entre los datos de toda la base de datos (BDD), dando nombre así a este tipo de BDD.

###### Ventajas de las Bases de Datos relacionales

Su uso es el más extendido y adaptado, por lo que los perfiles que las conocen son mayores y más baratos.

Tienen un mayor soporte y mejores suites de productos y add-ons para gestionarlas, debido al largo tiempo que llevan en el mercado.

La atomicidad de las operaciones en la base de datos. Esto es, que en estas bases de datos o se hace la operación entera o no se hace utilizando la famosa técnica del rollback.

Los datos deben cumplir requisitos de integridad tanto en tipo de dato como en compatibilidad.

###### Desventajas de las Bases de Datos relacionales

La atomicidad de las operaciones juega un papel crucial en el rendimiento de las bases de datos, haciendo que este tipo de BDD no siempre sea la que mejor rendimiento ofrezca.

Escalabilidad. Aunque está probada en muchos entornos productivos, por norma suele ser inferior a las bases de datos NoSQL.

##### 2. Bases de datos no relacionales (NoSQL)

###### Al contrario que las anteriores, en este tipo de almacenamiento de datos no se utilizan estructuras tan sencillas (tablas), ni se almacenan sus datos en forma de registros o campos.

###### Eso no significa que no utilicen lenguaje SQL.

###### Su propio nombre, como te explicábamos antes, no significa NO usar SQL, sino que significa no siempre usar SQL. Se apoyan en él, pero solo como un medio de apoyo, no como su herramienta de consulta principal.

###### Ventajas de una base de datos no relacional o NoSQL database

La escalabilidad y su carácter descentralizado. Soportan estructuras distribuidas.

Suelen ser bases de datos mucho más abiertas y flexibles. Permiten adaptarse a necesidades de proyectos mucho más fácilmente que los modelos de Entidad Relación.

Se pueden hacer cambios de los esquemas sin tener que parar bases de datos.

Escalabilidad horizontal: son capaces de crecer en número de máquinas, en lugar de tener que residir en grandes máquinas.

Se pueden ejecutar en máquinas con pocos recursos.

Optimización de consultas en base de datos para grandes cantidades de datos.

###### Desventajas de una base de datos NoSQL

No todas las bases de datos NoSQL contemplan la atomicidad de las instrucciones y la integridad de los datos. Soportan lo que se llama consistencia eventual.

Problemas de compatibilidad entre instrucciones SQL. Las nuevas bases de datos utilizan sus propias características en el lenguaje de consulta y no son 100% compatibles con el SQL de las bases de datos relacionales. El soporte a problemas con las queries de trabajo en una base de datos NoSQL es más complicado.

Falta de estandarización. Hay muchas bases de datos NoSQL y aún no hay un estándar como sí lo hay en las bases de datos relacionales. Se presume un futuro incierto en estas bases de datos.

Soporte multiplataforma. Aún quedan muchas mejoras en algunos sistemas para que soporten sistemas operativos que no sean Linux.

Poca usabilidad. Suelen tener herramientas de administración no muy usables o se accede por consola.

##### Cuándo utilizar bases de datos SQL y cuándo usar NoSQL

###### Esta serie de reglas te servirán de orientación sobre si necesitas una base de datos no relacional, o necesitas una base de datos relacional.

Cuando los datos deben ser consistentes sin dar posibilidad al error utilizar una base de datos relacional: SQL.

Cuando nuestro presupuesto no se puede permitir grandes máquinas y debe destinarse a máquinas de menor rendimiento: NoSQL.

Cuando las estructuras de datos que manejamos son variables: NoSQL database.

Análisis de grandes cantidades de datos en modo lectura: NoSQL.

Captura y procesado de eventos: NoSQL.

Tiendas online con motores de inteligencia complejos: NoSQL.

###### Es muy importante insistir en que, aunque parezca que en estos momentos deberías migrar tus bases de datos a NoSQL, tienes que tener muy en cuenta si las características de tu base de datos necesitan realmente una base de datos NoSQL o relacional antes de tomar esta decisión.

###### Aquí te dejamos este vídeo de la propia Google hablando sobre los dos tipos de base de datos. Es un vídeo antiguo en fecha, pero cuyo contenido sigue siendo totalmente válido:

## Work

### Sesión 2 - Introducción a Bases de Datos

#### Diseñando la base de datos

##### Para poder construir una base de datos es muy importante que información vamos a guardar en ella y de que forma vamos a estructurarla, de lo contrario podría resultar una base de datos con muchos errores que no funcione como esperamos. Es por esta razón que el proceso de diseño de bases de datos es de suma importancia. A continuación veremos algunas herramientas que nos ayudan en el proceso de diseño de una base de datos.

##### Ejemplo 1: Modelo Entidad Relación y Relacional

###### Desarrollo

Para este ejemplo veremos dos diagramas de gran utilidad en el proceso de diseño de una base de datos. Estos diagramas son el Modelo Entidad Relación y el Modelo Relacional. Para poder dibujarlos de forma sencilla se puede utilizar herramientas como draw.io

###### Modelo Entidad Relación

El modelo entidad-relación se basa en tres conceptos clave:

Entidad

Atributo

Relación.

Veamos con detalle cada uno de estos y como se adaptan a nuestro proyecto colaborativo.

###### Entidad

La entidad es la representación de un objeto que puede ser real o abstracto. Las entidades son esos modelos que identificamos en la sesión anterior, aquellos que representan actores de nuestro proyecto. Entonces nuestras entidades son:

Producto

Usuario

Venta

Reseña

En el diagrama una entidad se representa en forma de rectángulos con el nombre de la entidad, como los siguientes:

###### Atributo

Los atributos son los componentes o características que determina una entidad. Representan la información que deseamos guardad de cada entidad.

Por ejemplo, para la entidad producto tenemos los siguientes atributos:

Id

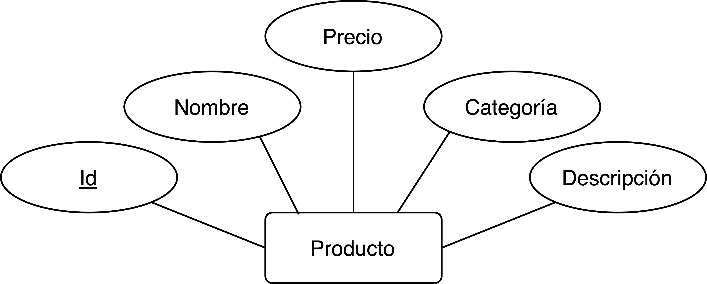
Nombre

Precio

Descripción

Categoría

En el diagrama los atributos tienen la forma de óvalo y estas conectados a la entidad sobre la cuál se definen, como se muestra a continuación:



El Id será el atributo con el cual identificaremos univocamente cada entidad y que mediante relaciones podremos acceder a la entidad. Este atributo se conoce como clave y se representa subrayando su nombre.

###### Relación

Una relación es una asociación entre varias entidades. Por ejemplo, las entidades producto y reseña están relacionadas y el nombre de esta relación es tiene

En un diagrama podemos observar una relación en forma de rombo y conectada a las entidades que relaciona:



Existen diferentes tipos de relaciones, estas se diferencian por su cardinalidad, es decir el número de registros que relacionan de cada una de las entidades. Y de esta forma tenemos 3 tipos de relaciones:

1:1

1:N o N:1

N:N

Nuestra relación entre producto y reseña es 1:N pues un producto tiene muchas reseñas pero cada reseña es exclusivamente sobre un producto.

###### Modelo Relacional

El modelo ER facilita las tareas de diseño conceptual de base de datos pero es necesario traducirlo a un esquema que sea compatible con un Sistema de Gestión de Base de Datos como lo es PostgreSQL.

El Modelo relacional es utilizado por la mayoría de los SGDB existentes en el mercado (Oracle, SQL Server, MaríaDB, SQLite).

En el modelo relacional se utiliza un grupo de tablas para representar los datos y las relaciones entre ellos. Cada tabla está compuesta por varias columnas que representan los atributos de la entidad y filas que serán registros o tuplas.

###### Restricciones de integridad

La integridad de los datos es la propiedad que asegura que información dada es correcta, al cumplir ciertas aserciones. Las restricciones de integridad son propiedades de la base de datos que se deben satisfacer en cualquier momento entre las cuales están:

Integridad de clave primaria

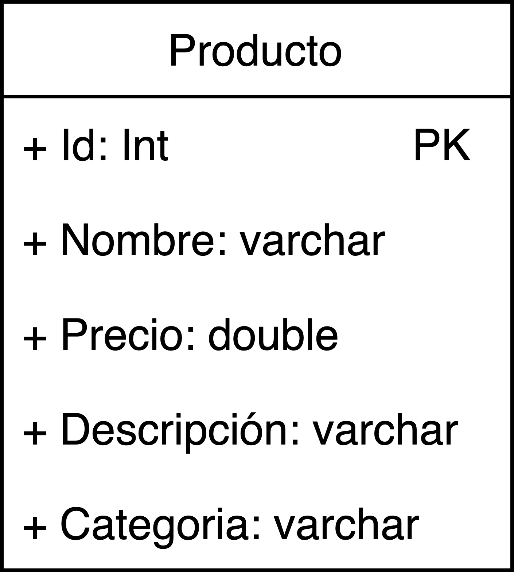
Integridad referencial

Tratamiento de valores nulos

Valores por defecto

###### ¿Cómo pasar del Modelo ER al Modelo Relacional?

Del ejemplo de la entidad "Producto"



Hemos hablado que el modelo relacional son tablas y columnas, en este ejemplo la entidad "Producto" será el nombre de nuestra tabla y cada atributo será una fila.

Como lo vimos anteriormente el uso de claves es importante ya que se utilizan para definir relaciones. Las tablas se relacionan mediante una "relación de clave primaria o de clave foránea", dónde:

Llave primaria: Es una columna o conjunto de columnas en una tabla cuyos valores identifican de forma exclusiva un registro de la tabla.

Llave foránea: Es una columna o conjunto de columnas en una tabla cuyos valores corresponden a los valores de la clave primaria de otra tabla.

##### Reto 1: Diseño de bases de datos

###### 

#### Sistemas gestores de bases de datos

##### Son programas que funcionan como interfaz entre lxs usuarixs y las bases de datos. Gestionando los datos, el motor de base de datos, y el esquema de la base para facilitar la organización y manipulación de los datos.En otras palabras, es un software mediante el cual nosotrxs como usuarixs podemos acceder a la base de daos y hacer consultas, manipular los datos, modificar el esquema y en general administrar la base de datos.

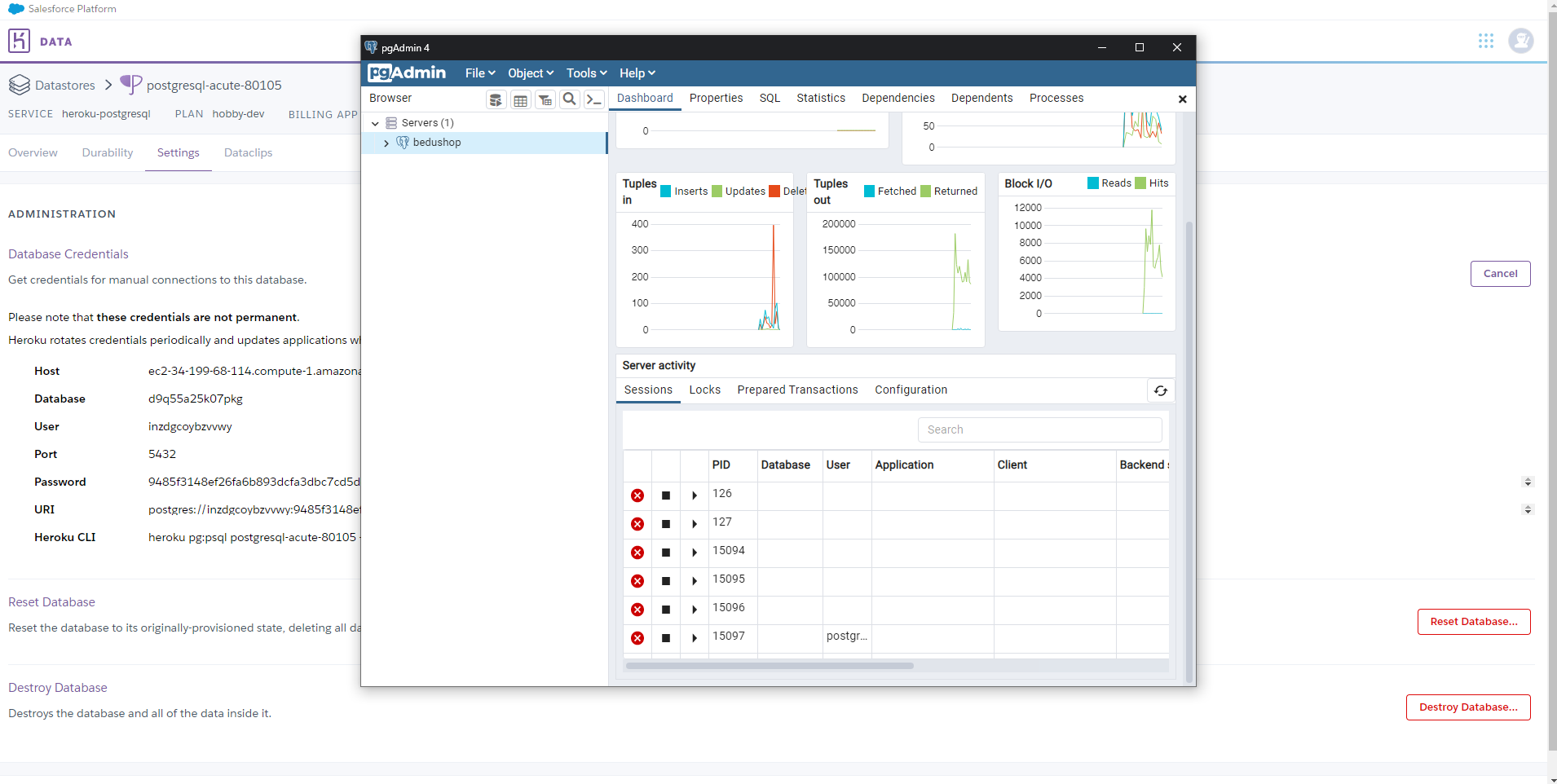
##### Para este modulo estaremos utilizando PostgreSQL como gestor de base de datos.

##### Ejemplo 2: Configuración de PostgreSQL

###### Ahora que ya tenemos el diseño de nuestra base de datos, vamos a convertirla en una base de datos real. Pero para eso necesitamos primero elegir el Sistema Gestor de Base de Datos que usaremos, para este modulo optamos por utilizar PostgreSQL, que es uno de los principales SGBD relacionales que se utilizan en la actualidad.

###### Pero el gestor no es suficiente, también es necesario un lugar en donde almacenar nuestra base de datos, para esto utilizaremos un servidor de bases de datos el cual hostearemos en Heroku, utilizando un add-on llamado Herroku Postgres.

###### En este ejemplo veremos el proceso de configuración del servidor y como conectarnos a él usando el cliente pgAdmin4



#### Creación de la base de datos

##### Ya que tenemos nuestro diseño y nuestro servidor de bases de datos, ahora vamos a crearla, es decir vamos a generar la estructura dentro del servidor. Y también vamos a comenzar a cargar datos a ella, para que nos demos una idea de la estructura que tendrá.

##### Ejemplo 3: Creación de la base de datos

###### En la barra lateral de pgAdmin abrimos Servers > bedushop > Databases > El nombre de la base de datos > Schemas

###### .

## Post-Work.