04

MongoDB - Spring Boot

Ciclo 4a:

Desarrollo de aplicaciones web

Objetivo de Aprendizaje

Identificar las principales **elementos** a utilizar para el desarrollo de los componentes **AccountDB** y **AccountMS**.

Parte 1

Bases de Datos: Definición

Una base de datos es una **colección de información** organizada, controlada por un **sistema gestor de base de datos** (Database Management System o **DBMS** por sus siglas en inglés), el cual funciona como una **interfaz** entre el cliente y la base de datos. Este se encarga de **administrar**, entre otras cosas, los datos, el motor de la base de datos y el esquema de la base de datos, para facilitar la organización y la manipulación de los datos.

[Imagen] Base de datos SQL. (s. f.). [PNG]. Hostgator. https://www.hostgator.mx/blog/wp-content/uploads/2019/11/bases-de-datos-sql-blog-v01.png

Bases de Datos: Tipos

Gracias a los **DBMS** es posible acceder, administrar, modificar, actualizar, controlar y organizar **fácilmente** los datos. Sin embargo, existen muchos **tipos** distintos de DBMS que se encargan de estas operaciones **a su manera**, siguiendo distintos **paradigmas** y **esquemas**.

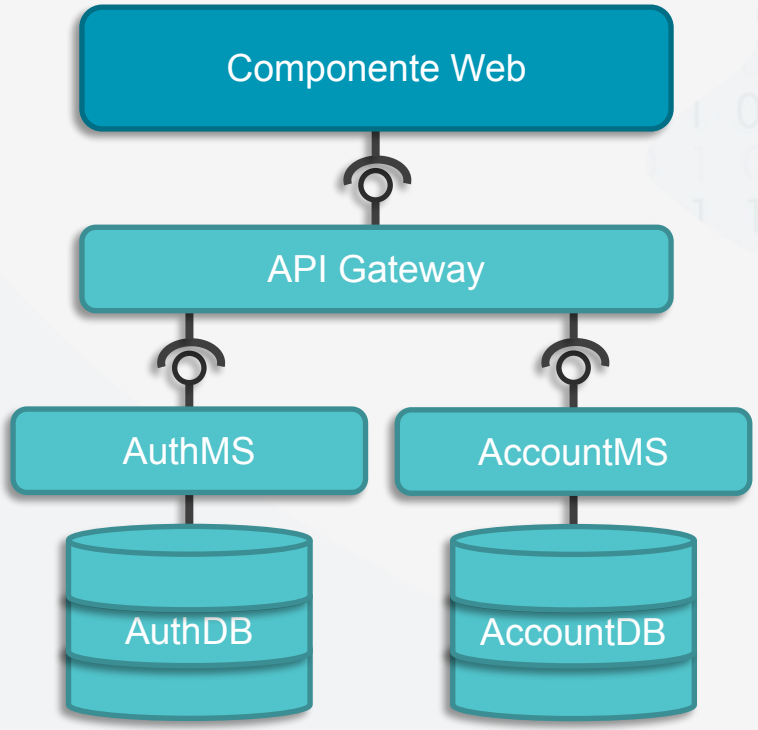
Así, el DBMS que utiliza una base de datos define su **tipo**, y por ende, el **motor** que utiliza, el **esquema** que sigue para almacenar los datos, el **lenguaje de dominio específico** que se debe utilizar para dar instrucciones a la base de datos, entre otras cosas.

[Imagen] DBMS. (s. f.). [PNG]. LearnSQL. https://learnsql.com/blog/what-is-dbms/DBMS.png

Bases de Datos: Tipos

Actualmente se tienen **dos** principales **tipos** de bases de datos, las bases de datos **relacionales** y las bases de datos **NoSQL**.

En el sistema de software planteado, ya se ha trabajado con una base de datos relacional. **AuthDB** usa el motor **PostgreSQL**, y ahora se trabajará con una base de datos NoSQL. **AccountDB** usará el motor **MongoDB**.



El futuro digital es de todos

MinTIC

Bases de Datos Relacionales

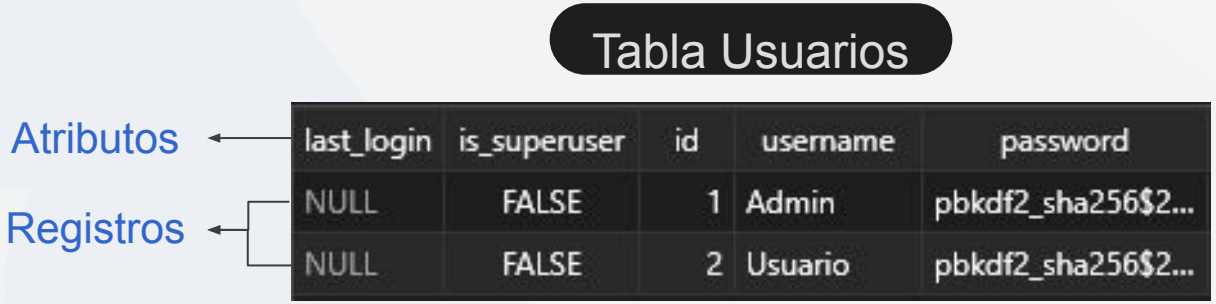
A continuación, se recordarán algunos **conceptos** claves de las bases de datos **relacionales**, estos permitirán entender la **diferencia** con las bases de datos **NoSQL**.

UNIVERSIDAD NACIONAL DE COLOMBIA

MisionTIC 2022

Relacionales: Definición

Las bases de datos relacionales son aquellas cuyo DBMS sigue un **paradigma** basado en **relaciones**. Es decir, que organizan los datos en una serie de **tablas** con **filas** y **columnas**, donde cada columna es un **atributo** de la tabla, y cada fila es un **registro**, y se establecen relaciones o **referencias** entre atributos de distintas tablas. Se puede comparar con una tabla de Excel, sin embargo, una base de datos relacional es mucho más **compleja**.



Relacionales: Lenguaje SQL

Generalmente, las bases de datos relacionales utilizan el **lenguaje de consulta** estructurado **SQL** para registrar y consultar datos.

Algunos de los DBMS más famosos que siguen este paradigma son **PostgreSQL**, MySQL, MariaDB y Microsoft SQL Server.

[Imagen] Bases de datos relacionales. (s. f.). [PNG]. Diego Calvo. https://www.diegocalvo.es/wp-content/uploads/2017/11/Base-de-Datos-Relacionales.png

El futuro digital es de todos

MinTIC

Bases de Datos NoSQL

A continuación, se explicarán algunos **conceptos** claves para **trabajar** con bases de datos **NoSQL**.

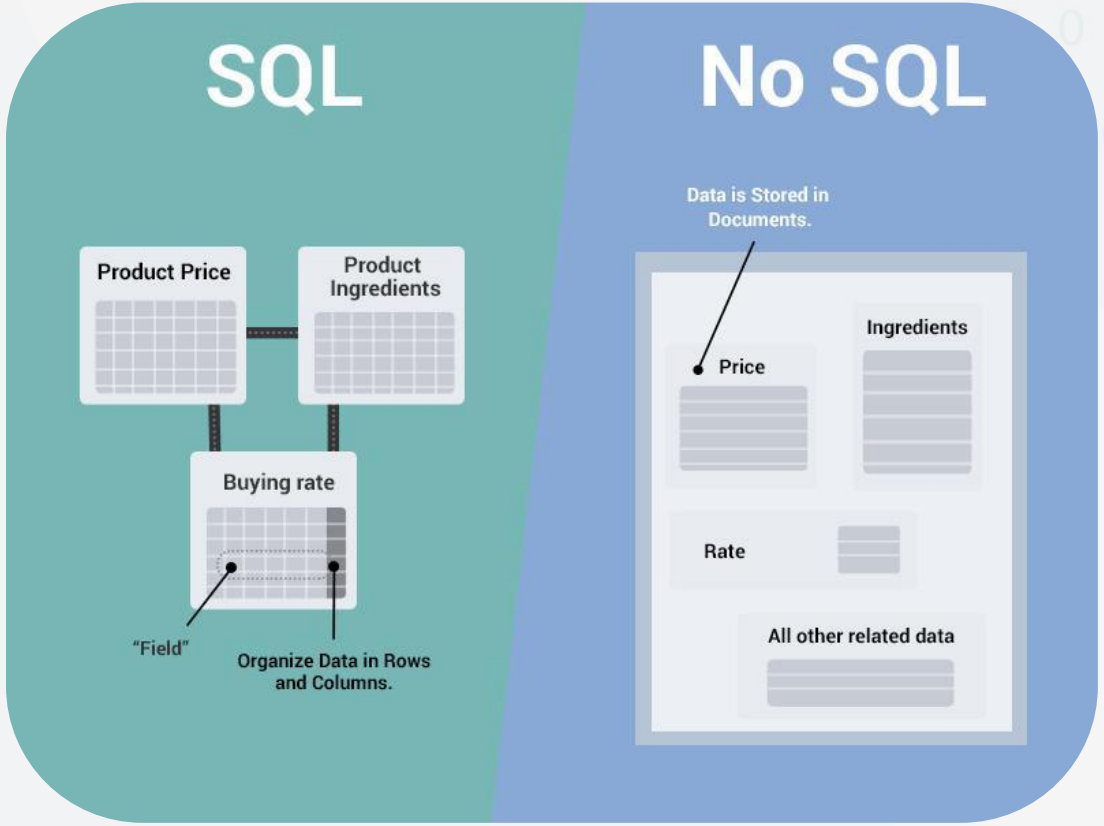
Universidad Nacional de Colombia

MisiónTIC 2022

NoSQL: Definición

Las bases de datos NoSQL se refieren a todas aquellas que **no siguen el paradigma relacional**.

Existen varios subtipos como las bases de datos basadas en grafos, de columna amplia y las de llave-valor. Sin embargo, uno de los más utilizados es el **basado en documento** en el cual los datos se almacenan como un **objeto** o un **documento** de tipo **JSON**.



[Imagen] SQL-vs-NoSQL. (s. f.). [JPG]. Wpengine. https://ucppr2k2q2u3lbr9b2ah3a31-wpengine.netdna-ssl.com/blog/wp-content/uploads/sites/3/2019/09/SQL-vs-NoSQL.jpg

El futuro digital es de todos

MinTIC

NoSQL: JSON

El uso de documentos como JSON ofrece una mayor **flexibilidad** para el almacenamiento de los datos, **reduce la duplicación** de los datos, facilita la **escalabilidad** y el rápido **despliegue**.

Estos documentos pueden estar en formato XML o JSON, siendo este último el más utilizado por su **facilidad de lectura**. En la imagen se puede ver un ejemplo de **documento JSON**.

[Imagen] JSON. (s. f.). [PNG]. MongoDB. https://webassets.mongodb.com/_com_assets/cms/1-lwnlfl1ryn.png

UNIVERSIDAD NACIONAL DE COLOMBIA

MisiónTIC 2022

El futuro digital es de todos

MinTIC

NoSQL: Documentales

Algunos de los DBMS más utilizados son **MongoDB**, CouchDB, ArangoDB y ElasticSearch. Si bien algunas bases de datos **NoSQL** pueden utilizar el lenguaje **SQL**, generalmente cada DBMS crea su propio lenguaje para registrar y consultar datos. De igual forma, existen **interfaces gráficas** de usuario para la **manipulación y administración** de las bases de datos sin necesidad de conocer el lenguaje.

Universidad Nacional de Colombia

MisiónTIC 2022

NoSQL: MongoDB

Uno de los DBMS de tipo documental más utilizado es **MongoDB**, este usa archivos de tipo **BSON** para almacenar la información, estos archivos son muy similares a los archivos **JSON** pero con algunas propiedades adicionales.

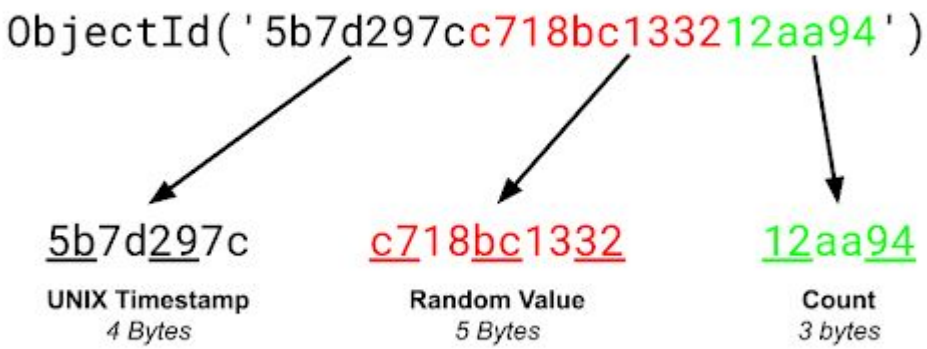
MongoDB es desarrollado por **MongoDB** Inc., esta corporación ofrece muchas **herramientas** que facilitan el uso de MongoDB. Una de las herramientas más famosas es **Atlas**, esta permite **desplegar** una base de datos de manera fácil y rápida.

[Imagen] Atlas Logo. (s. f.). [Ilustración]. https://q.foolcdn.com/image/?url=https%3A%2F%2Fq.foolcdn.com%2Feditorial%2Fimages%2F635884%2Fatlas_icon_blk_stackedlarge.png&w=1200&op=resize
[Imagen] MongoDB Logo. (s. f.). [Gráfico]. https://infinapps.com/wp-content/uploads/2018/10/mongodb-logo.png

MongoDB: Documentos BSON

La principal funcionalidad adicional que ofrecen los documentos **BSON** es el **ObjectId**, este es un campo que poseen todos los documentos y permite que **MongoDB** los **identifique**. Este valor solo tiene sentido en el contexto de la base de datos, en otro escenario es un simple String.

BSON también ofrece algunos **tipos** de **datos adicionales**, que normalmente no están soportados en los archivos JSON.

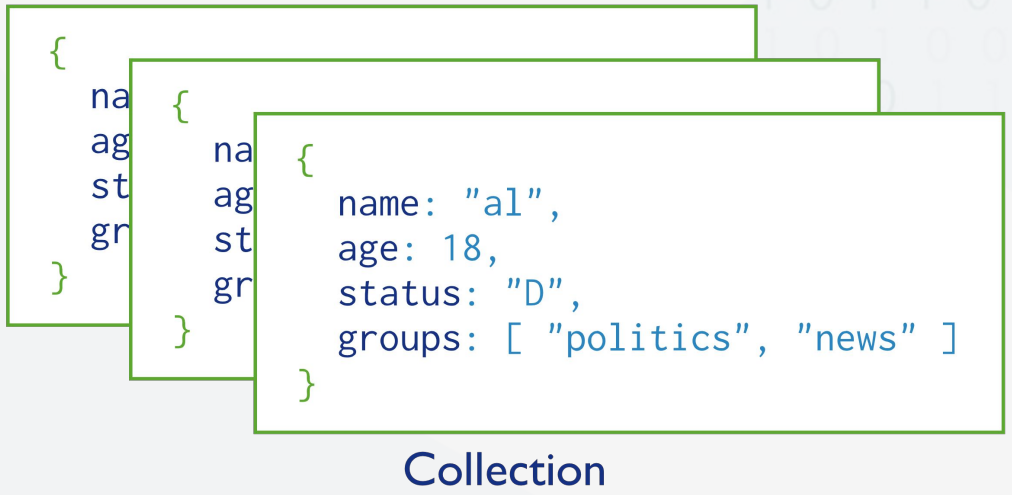


[Imagen] ObjectId. (s. f.). [Gráfico]. https://www.mongodb.com/developer/images/objectid.png

MongoDB: Colecciones

Dentro de **MongoDB** cada **registro** es almacenado en un **documento**, por ejemplo al guardar la información del cliente #023 se creará un documento únicamente para ese registro. Como es de esperarse pueden existir **muchos documentos** con estructuras similares, y en algunos momentos puede ser necesario realizar operaciones sobre estos (por ejemplo, obtener todos los clientes).

Para esto **MongoDB** ofrece el concepto de **colección**, básicamente una colección **agrupa documentos** y permite realizar **operaciones grupales** sobre estos.



Collection

[Imagen] Collection. (s. f.). [Gráfico]. https://docs.mongodb.com/manual/images/crud-annotated-collection.bakedsvg.svg

MongoDB: Sub-documentos

La principal ventaja de trabajar con **MongoDB** es la flexibilidad, y esto se ve reflejado en los **subd-ocumentos**. Un sub-documento es un **documento embebido** dentro de otro documento, este patrón resulta bastante útil en muchos escenarios, ya que en lugar de tener varios campos sueltos o un documento externo, se pueden agrupar en una estructura más clara.

Se debe hacer uso de esta técnica únicamente cuando el **sub-documento** esté **estrechamente relacionado** con el documento.

```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
             phone: "123-456-7890",
             email: "xyz@example.com"
           },
  access: {
             level: 5,
             group: "dev"
           }
}
```

Embedded sub-document

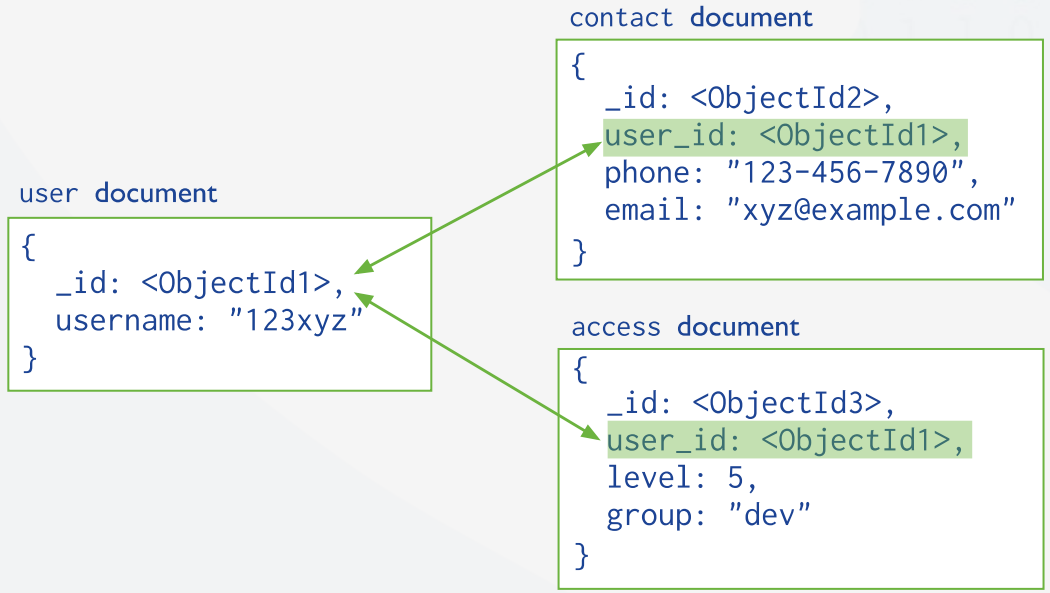
Embedded sub-document

[Imagen] Subdocumentos. (s. f.). [Gráfico]. https://docs.mongodb.com/manual/images/data-model-denormalized.bakedsvg.svg

MongoDB: Referencias

Cuando no existe una relación estrecha entre dos documentos, se puede hacer uso de **referencias**. Una referencia es un **campo** cuyo **valor** es un **apuntador** (con ayuda del **ObjectId**) a otro **documento**.

Este concepto es muy similar a las **relaciones** de las **bases** de **datos relacionales**. De hecho, con referencias se pueden plantear relaciones **1:1, 1:N o M:N**. Por esto MongoDB se denomina **NoSQL** (Not Only SQL), es decir que ofrece los mismo que SQL y mucho más.



[Imagen] MongoDB References. (s. f.). [Gráfico]. https://docs.mongodb.com/manual/images/data-model-normalized.bakedsvg.svg

Parte 2

El futuro digital es de todos

MinTIC

SPRING BOOT

Para hacer uso del framework Spring Boot es necesario conocer algunos conceptos.

UNIVERSIDAD NACIONAL DE COLOMBIA

MisiónTIC 2022

Lenguaje: Java

Java es un **lenguaje** de **programación** de propósito general **orientado** a **objetos**, el cual apareció por primera vez en **1995.** Java es rápido, seguro y confiable.

Para **ejecutar** aplicaciones construidas en **Java** solamente en necesario un **JRE (Java Runtime Environment).** Pero para **construir aplicaciones** se necesita un **JDK (Java Developer Kit).**

El futuro digital es de todos

MinTIC

Maven

Maven es una **herramienta** open-source creada en 2001, su principal función es **simplificar** los **procesos** de **construcción** de **aplicaciones**, esto incluye **compilación** y **ejecución**.

Para hacer **uso** de **Maven** es necesario tener un **JDK** e instalar un **paquete adicional** (proveído por **Apache Maven**).

[Imagen] Maven Logo. (s. f.). [Ilustración]. https://3.bp.blogspot.com/-YoXSXqqOyvI/WXsLGwITvyI/AAAAAAAAIFQ/hwobCq7-OKYvqI3FJe4dRh0NFz2ImJKiwCEwYBhgL/s1600/maven%2Bjava%2Blogo.png

UNIVERSIDAD NACIONAL DE COLOMBIA

MisiónTIC2022

Framework: Spring Boot

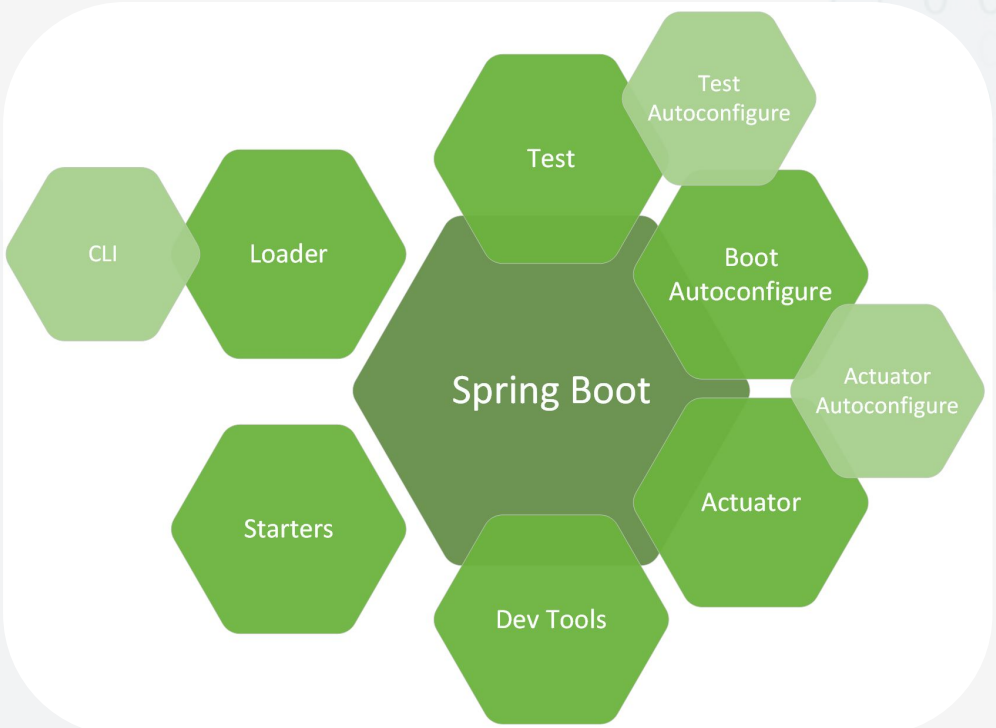
Spring Boot es un **framework** que nace a partir de **Spring Core**, su objetivo es **simplificar** la **construcción** de **aplicaciones** haciendo que el **desarrollador** se centre **únicamente** en la **solución, omitiendo** los problemas de **construcción**.

Para lograrlo, provee un **sistema de dependencias** y un **inicializador** que en **cuestión** de **segundos** es capaz de generar un **proyecto** listo para **desarrollar**, sin ninguna **configuración adicional**.

[Imagen] Spring Boot Logo. (s. f.). [Ilustración]. https://niixer.com/wp-content/uploads/2020/11/spring-boot.png

Dependencias

El paquete inicial de **Spring Boot** es supremamente **ligero** y provee **pocas funcionalidades**, pero está **diseñado** para crecer de manera **automática** y **genérica;** es decir, cuando se necesita **incluir** una **funcionalidad** a la aplicación (por ejemplo conectarse a una base de datos), se **agrega** una **nueva dependencia** y Spring Boot la **integra** sin necesidad de **configuraciones** adicionales.



[Imagen] Spring Boot Dependencias. (s. f.). [Ilustración]. https://pradeepmeruva.com/assets/img/post/module_structure.png

IDE: IntelliJ IDEA

Un **IDE** (Integrated Development Environment) es una **herramienta** que integra **distintos servicios** que **facilitan** el **desarrollo** de una aplicación.

Uno de los **IDEs** más populares para **Java** es **IntelliJ IDEA**. Este **integra** algunas herramientas **básicas** como **editor**, **terminal** y algunas más **complejas** como el **manejo de dependencias**.

[Imagen] IntelliJ IDEA Logo. (s. f.). [Ilustración]. http://assets.stickpng.com/images/58480910cef1014c0b5e48f7.png

Parte 3

El futuro digital es de todos

MinTIC

Desarrollo del Microservicio

Para implementar el microservicio es necesario considerar algunos conceptos básicos con respecto al funcionamiento de Spring Boot.

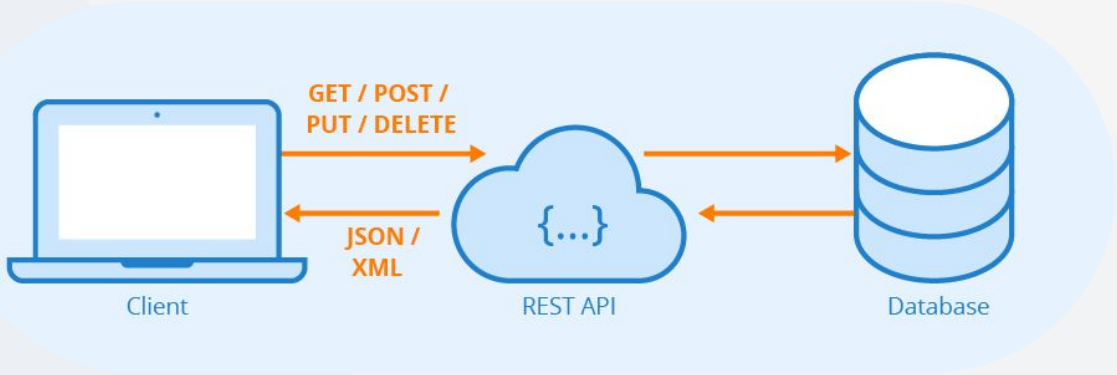
UNIVERSIDAD NACIONAL DE COLOMBIA

MisiónTIC 2022

Microservicio: API-REST

El **microservicio** a desarrollar expone una **API-REST**, la cual se basa en una **comunicación cliente-servidor sin estado**; es decir, que el microservicio únicamente responderá a las peticiones HTTP (**GET**, **POST**, **PUT** y **DELETE**), y no almacenará metadatos de sus clientes.

Estas características permiten que el **microservicio** tenga una **interfaz uniforme**, que lo hará **ligero** y **compatible** con los demás componentes.



[Imagen] API REST. (s. f.). [Ilustración]. https://www.astera.com/wp-content/uploads/2020/01/rest.png

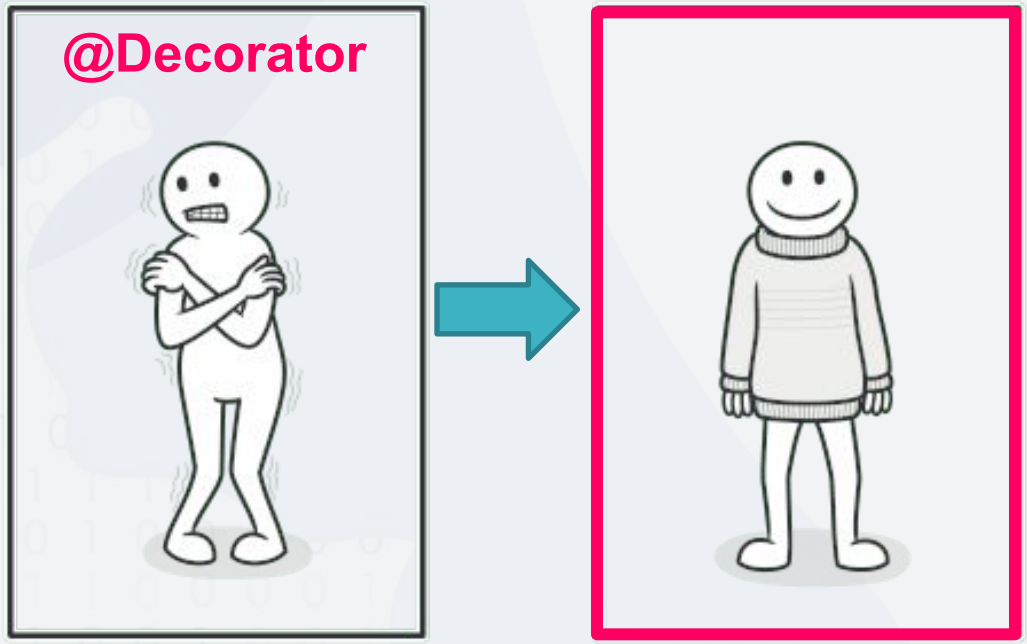
Spring Boot: Modelo

En **Spring Boot**, los **Modelos** se representan **definiendo** una serie de **entidades**, las cuales son una **abstracción** de los **datos** que se están **usando** dentro de la **aplicación**.

En el entorno de **Java**, la abstracción en **entidades** permite manejar a los **datos** como **objetos** y no como **datos independientes**. Por **ejemplo**, en lugar de tener algunas **variables** con la información del **usuario**, se tiene una **Entidad** o **Modelo** llamada **Usuario** con toda la información.

[Imagen] Data Representantion. (s. f.). [Fotografía]. https://www.lawebdelprogramador.com/usr/211000/211493/profileImage_origin.jpg?1486135451

Spring Boot: Decorador



Un **decorador** es una **anotación**, que se agrega antes de un **método**, **atributo**, **clase** o **interfaz**. Estos siempre inician con una arroba (**@**), por ejemplo **@RestController**.

Spring Boot usa estas **anotaciones** para detectar **componentes** o **propiedades**, y de esta manera **añadir funcionalidades** extras o **manejarlos** de manera **diferente**.

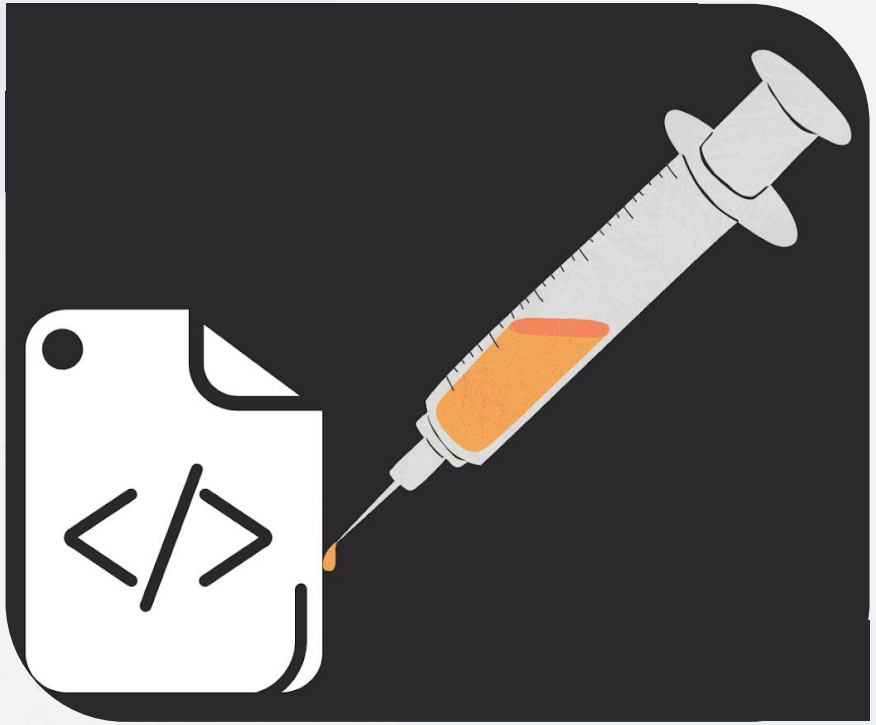
Spring Boot: Excepciones

Cuando ocurre un **error** dentro de **Java**, se **genera** una **excepción**. De **cara** al **desarrollador** estas son **útiles** y **brindan** información **importante**, pero para el **cliente final** son **ambiguas** y poco claras.

Para **solucionar** lo anterior, **Spring Boot** permite manejar **excepciones personalizadas** y **controlarlas**, **usando** un concepto llamado **Advices,** los cuales **transforman** las **excepciones** en **mensajes claros** para el **cliente**.

[Imagen] Error Wallpaper. (s. f.). [Fotografía]. https://image.shutterstock.com/image-photo/computer-error-message-on-screen-260nw-1703102707.jpg

Spring Boot: Inyección



Dentro de los **controladores** u **otros componentes**, en muchas ocasiones se **necesitan** algunas **instancias** de otras **clases**, para su funcionamiento. A estas se les **conocen** como **dependencias**.

En un **entorno** de trabajo con **Java**, dichas **instancias** deben ser **construidas** y **agregadas** donde son requeridas, pero en **Spring Boot** no es necesario realizar este proceso, ya que Spring Boot se **encarga** de **todo**. Este **mecanismo** se conoce como **Inyección de Dependencias**.

[Imagen] Inyección de Dependencias. (s. f.). [Ilustración]. https://1.bp.blogspot.com/-uq1dda8x5IM/X3EX4N56ITI/AAAAAAABJ04/SE-VENo_mWMISPv5ZqJzWvY0UmpZumwcgCLcBGAsYHQ/s2048/Copy%2Bof%2BHow%2Bto%2Buse%2BSets%2Bin%2BSw%2B%25283%2529.png