autentia

Frameworks de desarrollo

Quien soy?

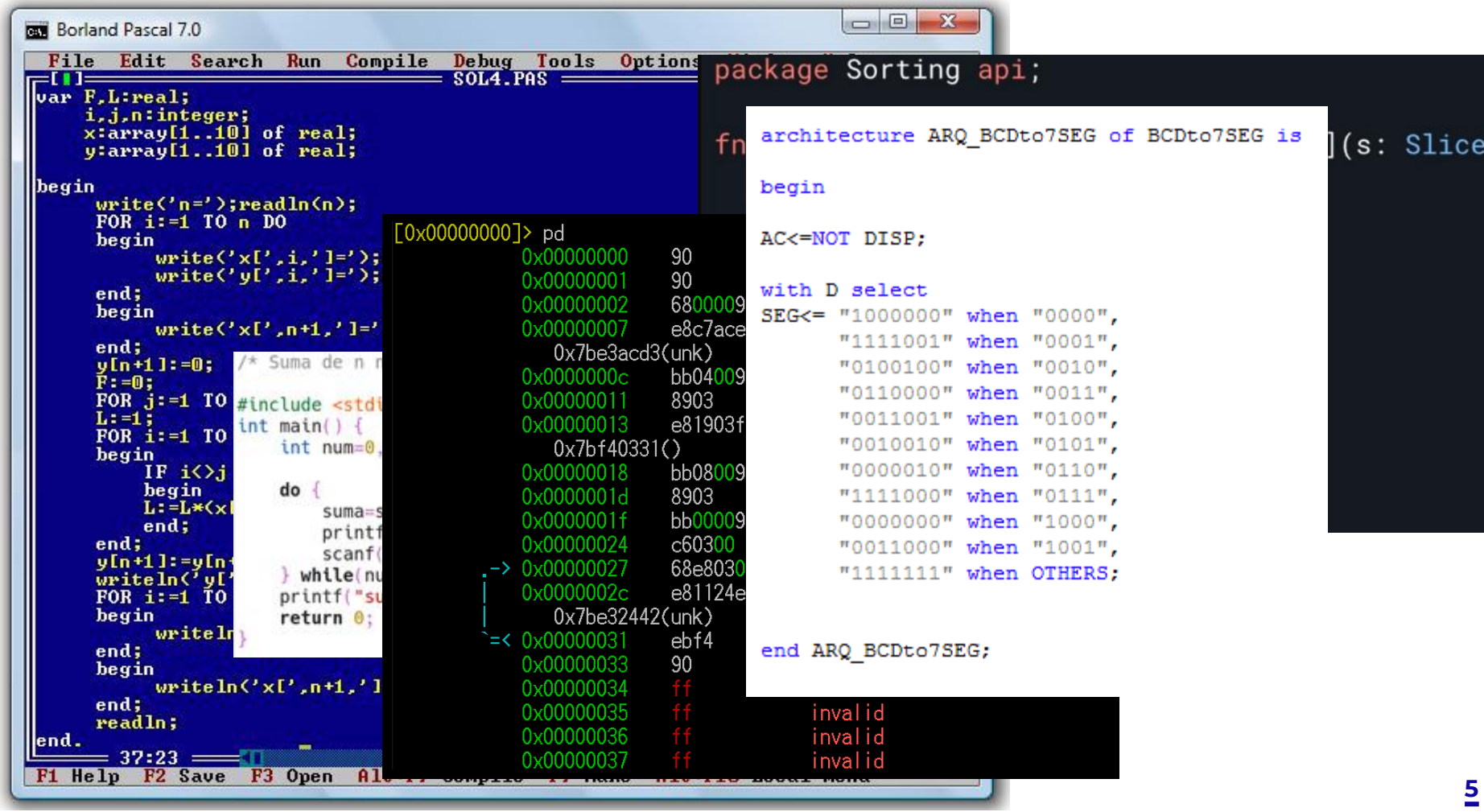
- Arquitecto software en Autentia
- Banca, Logística, Aerolíneas, ...
- Arquitecturas de microservicios, REST, ...
- Metodologías agile

@ifdezmolina

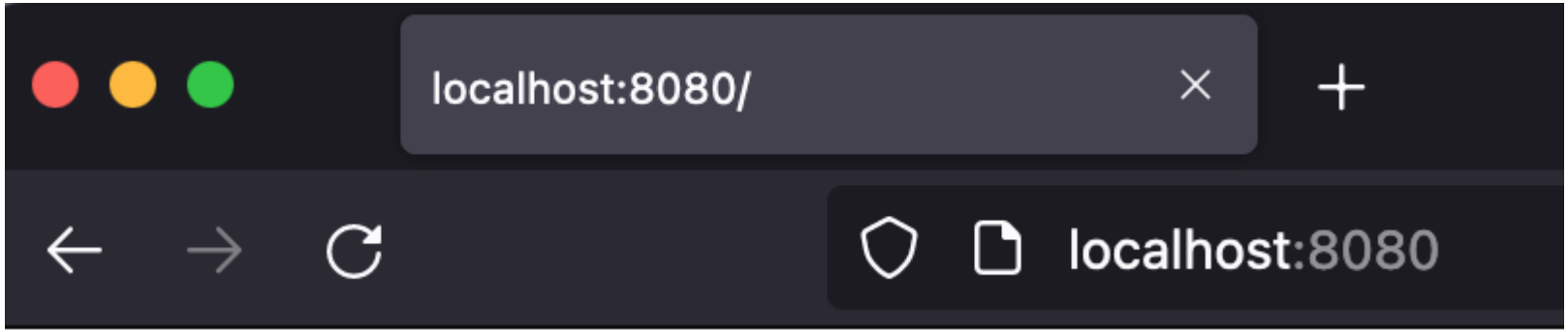
https://www.linkedin.com/in/ismael-fernandez-molina-7579b811

ifernandez@autentia.com

Comencemos...


Borland Pascal 7.0
File Edit Search Run Compile Debug Tools Options
SOL4.PAS
var F,L:real;
i,j,n:integer;
x:array[1..10] of real;
y:array[1..10] of real;
begin
write('n=');readln(n);
FOR i:=1 TO n DO
begin
write('x[',i,']=');
write('y[',i,']=');
end;
begin
write('x[',n+1,']='
end;
y[n+1]:=0;
F:=0;
FOR j:=1 TO
L:=1;
FOR i:=1 TO
begin
IF i<>j
begin
L:=L*(x
end;
end;
y[n+1]:=y[n
writeln('y[
FOR i:=1 TO
begin
writeln
end;
begin
writeln('x[',n+1,']
end;
readln;
end.
37:23
F1 Help F2 Save F3 Open
/* Suma de n
#include <stdi
int main() {
int num=0,
do {
suma=s
printf
scanf(
} while(nu
printf("su
return 0;
}
[0x00000000]> pd
0x00000000 90
0x00000001 90
0x00000002 6800009
0x00000007 e8c7ace
0x7be3acd3(unk)
0x0000000c bb04009
0x00000011 8903
0x00000013 e81903f
0x7bf40331()
0x00000018 bb08009
0x0000001d 8903
0x0000001f bb00009
0x00000024 c60300
.-> 0x00000027 68e8030
| 0x0000002c e81124e
| 0x7be32442(unk)
`=< 0x00000031 ebf4
0x00000033 90
0x00000034 ff
0x00000035 ff invalid
0x00000036 ff invalid
0x00000037 ff invalid
package Sorting api;
fn](s: Slice(T
architecture ARQ_BCDto7SEG of BCDto7SEG is
begin
AC<=NOT DISP;
with D select
SEG<= "1000000" when "0000",
"1111001" when "0001",
"0100100" when "0010",
"0110000" when "0011",
"0011001" when "0100",
"0010010" when "0101",
"0000010" when "0110",
"1111000" when "0111",
"0000000" when "1000",
"0011000" when "1001",
"1111111" when OTHERS;
end ARQ_BCDto7SEG;

```xml
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
          http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">

    <display-name>Helloworld application</display-name>

    <servlet>
        <servlet-name>HelloWorld</servlet-name>
        <servlet-class>es.uah.hello.HelloWorldServlet</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>HelloWorld</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>

</web-app>
```



Hello World!

Problemas...

- Demasiada configuración
- Tareas muy manuales
- Facilidad para cometer errores
- Mucho trabajo en comparación al resultado obtenido
- En definitiva...

Soluciones

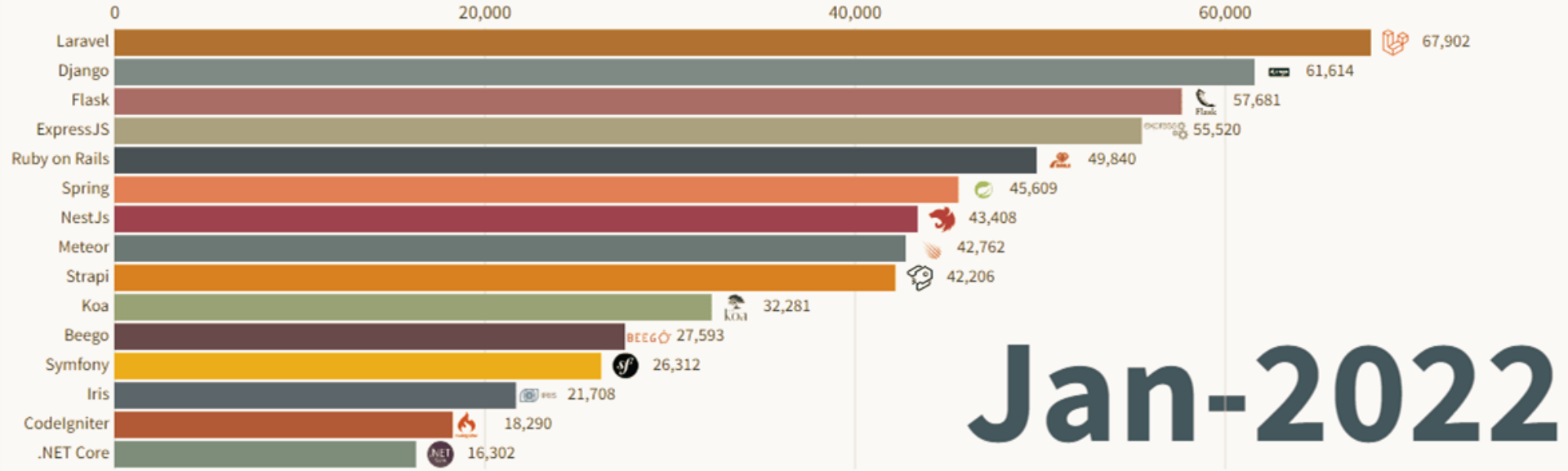
- Buenas prácticas
- Código de calidad: SOLID
- Testing: TDD
- Patrones de diseño
- Frameworks de desarrollo

Framework

“Un framework es un marco o esquema de trabajo generalmente utilizado por programadores para realizar el desarrollo de software. Utilizar un framework permite agilizar los procesos de desarrollo ya que evita tener que escribir código de forma repetitiva, asegura unas buenas prácticas y la consistencia del código.”

Cuantos hay?

Most Popular Backend Frameworks



SIMFORM

TOP 7 FRONTEND JAVASCRIPT FRAMEWORKS

REACT

VUE.JS

ANGULAR

jQUERY

SVELTE

EMBER

BACKBONE


General Registrat
WHICH ONE?

Depende

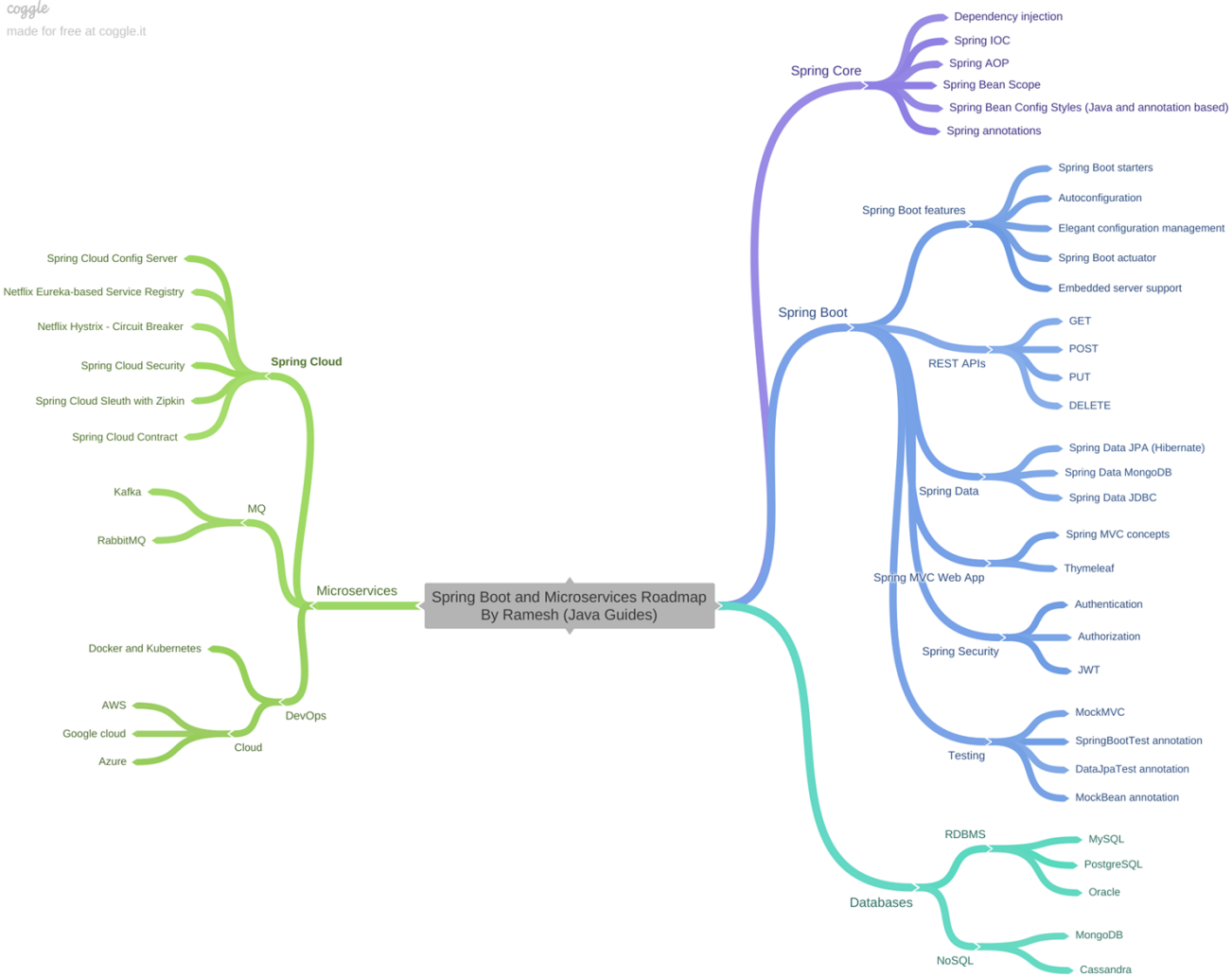

spring

Spring

- Framework más popular para el desarrollo de aplicaciones en Java.
- Nace con el objetivo de unificar y facilitar la construcción de aplicaciones en la plataforma Java, incluyendo soporte para Kotlin o Groovy.
- Framework modular. Nos permite utilizar sólo aquellos módulos que realmente necesitamos.
- El ecosistema de Spring es muy extenso, (**más de 20 proyectos**), de los cuales los más conocidos son **Spring Boot** y **Spring Data**.

Spring

- Característica principal: el contenedor de **Inversion of Control (IoC)**
- Soporta gestión de **transacciones** declarativas
- Acceso remoto mediante **RMI** o **servicios Web** y varias opciones para persistir sus datos.
- Ofrece un framework **MVC** (Model View Controller) completo
- Permite integrar **AOP** (Aspect Oriented Programming) de forma transparente al desarrollo
- Permite a los desarrolladores centrarse principalmente en la lógica de negocio a la hora de construir aplicaciones mientras que él asume el peso de las piezas de infraestructura



Spring IoC

- **Inversión de control (IoC)** se utiliza en el diseño orientado a objetos para delegar en un tercero diferentes tipos de flujos de control para lograr un bajo acoplamiento → me despreocupo por la creación de clases terceras y delegamos en el tercero su propia creación
- Uno de los patrones que implementan la inversión de control es la **Inyección de Dependencias**, de modo que la creación y establecimiento de las dependencias de una clase es controlada por un framework o contenedor
- Reduce el acoplamiento
- Aumenta la modularidad y extensibilidad
- Mejora la testeabilidad

Spring IoC

- El contenedor (Spring IoC container) es el que asume la responsabilidad de la gestión del ciclo de vida de las dependencias, conocidas como Beans, y de su inyección en aquellas clases que las necesiten
- Dicha inyección se realiza normalmente a través del constructor,

Spring IoC

```
public class PlayersServlet extends HttpServlet{

    private final PlayerRepository playerRepository;

    public PlayersServlet() throws SQLException, ClassNotFoundException{
        super();
        playerRepository = new PostgresPlayerRepository();
    }
```

Spring IoC

```
public class PostgresPlayerRepository implements PlayerRepository{

    private static final String DATABASE_URL = "jdbc:postgresql://localhost:5432/postgres";
    private static final String DATABASE_USER = "postgres";
    private static final String DATABASE_PASSWORD = "postgres";

    private final Connection con;

    public PostgresPlayerRepository() throws ClassNotFoundException, SQLException{
        //load Postgres driver
        Class.forName("org.postgresql.Driver");
        con = getConnection(DATABASE_URL, DATABASE_USER, DATABASE_PASSWORD);
        System.out.println("--> Connection with database stablished");
    }
```

Spring IoC

```
@WebServlet(name = "PlayersServlet",urlPatterns = "/players")
public class PlayersServlet extends HttpServlet{

    private final PlayerRepository playerRepository;

    @Autowired
    public PlayersServlet(PlayerRepository playerRepository) {
        super();
        playerRepository = playerRepository;
    }
```

Spring

```
@Repository
public class PostgresPlayerRepository implements PlayerRepository{

    private final JdbcTemplate jdbcTemplate

    @Autowired
    public PostgresPlayerRepository(JdbcTemplate jdbcTemplate){
        this.jdbcTemplate = jdbcTemplate;
    }
```

Spring - contexto

A partir de las clases de la aplicación y una configuración declarada en XML, Java o anotaciones, se crea e inicializa la instancia del ApplicationContext que gestionará el ciclo de vida los beans definidos en nuestra aplicación

Spring - Beans

- Cualquier objeto cuyo ciclo de vida lo gestiona el contenedor de dependencias

Spring - Configuración

- Antes de Spring 3.0
- En desuso a favor del uso de anotaciones y configuraciones programáticas

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">

    <bean id="teacher" class="com.autentia.Teacher"></bean>
    <bean id="course" class="com.autentia.Course">
        <property name="teacher" ref="teacher"></property>
    </bean>
</beans>
```

Spring - Anotaciones

- **@ComponentScan**: Indica los paquetes que deben escanearse para identificar las clases etiquetadas como beans.

```
@Configuration
@ComponentScan(basePackage = com.autentia.courses)
class AppConfig {}
```

Spring - Anotaciones

- **@Component**: anotación general para indicar que una clase es un bean
- **@Controller, @Service, @Repository**: a efectos de identificación de bean es lo mismo que @Component. Permiten clasificar nuestros beans según el contexto de la clase: clases de controlador, de capa de servicio o capa DAO/repositorio.
- **@Autowired**: usada a nivel de constructor o propiedad de la clase, para indicar que Spring debe ser el encargado de inyectar esos beans como dependencia.

Spring - Resolución de conflictos

Cuando tenemos múltiples implementaciones sobre una misma interfaz, al usar la anotación @Autowired, Spring tendrá problemas para identificar qué bean queremos

```
Field “campo anotado con @Autowired” in ... required a single bean,
but 2 were found
```

Spring - Resolución de conflictos

```
public interface CarService { ... }

@Service
public class ElectricCarService implements CarService { ... }

@Service
public class HybridCarService implements CarService { ... }
```

Spring - Resolución de conflictos

```
@Controller
public class CarController {

      @Autowired
      private CarService carService;

      ...
}
```

Spring - Resolución de conflictos

- **@Qualifier**: se usa junto con la anotación **@Autowired** y se pasa entre paréntesis el nombre de la implementación que queremos inyectar.

```
@Autowired
@Qualifier("hybridCarService")
private CarService carService;
```

Spring - Resolución de conflictos

- **@Primary**: para indicar que cuando no se especifica el bean a inyectar, este se debe usar por defecto

```
@Service
@Primary
public class ElectricCarService implements CarService { ... }
```

Spring Data

- Unificar y facilitar el acceso a tecnologías de acceso a datos, tanto a bases de datos relacionales como a NoSQL así como a servicios basados en la nube, etc
- Dividido en submódulos en función de la tecnología
- Spring Data JDBC
- Spring Data JPA.
- Spring Data Rest
- Spring Data Redis
- Spring Data for Apache Cassandra
- Spring Data Elasticsearch
- ...

JDB

- AP
- Fu
  op
  eje

ría de

nes,

```
public List < User > findAll() throws SQLException {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection(
            "jdbc:mysql://localhost:3306/db_name", "db_username", "db_password");
        Statement stmt = con.createStatement();
        ResultSet result = stmt.executeQuery("select * from users");
        List < User > users = new List()
        while (result.next()) {
            Long id = result.getLong("id");
            String firstName = result.getString("first_name");
            String lastName = result.getString("last_name");

            // Construimos el objeto User
            User user = new User(id, firstName, lastName);
            users.add(user);
        }
        con.close();
        return users
    } catch (SQLException e) {
        System.out.println(e);
    }
}
```

Spring JDBC

Abstracción sobre JDBC para preocuparnos sólo de:

- Definir los parámetros de conexión
- Especificar la operación (consulta, actualización, etc.)
- Realizar alguna tarea con esos resultados.

Spring se encargará del resto: abrir y cerrar la conexión a la base de datos, gestionar las excepciones, iterar sobre los resultados, etc

Spring JDBC

```
## MySQL
spring.datasource.url=jdbc:mysql://localhost:3306/db_name
spring.datasource.username=db_username
spring.datasource.password=db_password
```

```
public class User {
    private Long id;
    private String firstName;
    private String lastName;
    // constructors, getters, setters
}
```

Spring JDBC

```
@Repository
public class JdbcUserRepository {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    @Override
    public List<User> findAll() {
        return jdbcTemplate.query(
                "select * from users",
                (rs, rowNum) ->
                        new User(
                                rs.getLong("id"),
                                rs.getString("first_name"),
                                rs.getString("last_name")
```

Spring Data JDBC

Abstracción de Spring Data sobre Spring JDBC:

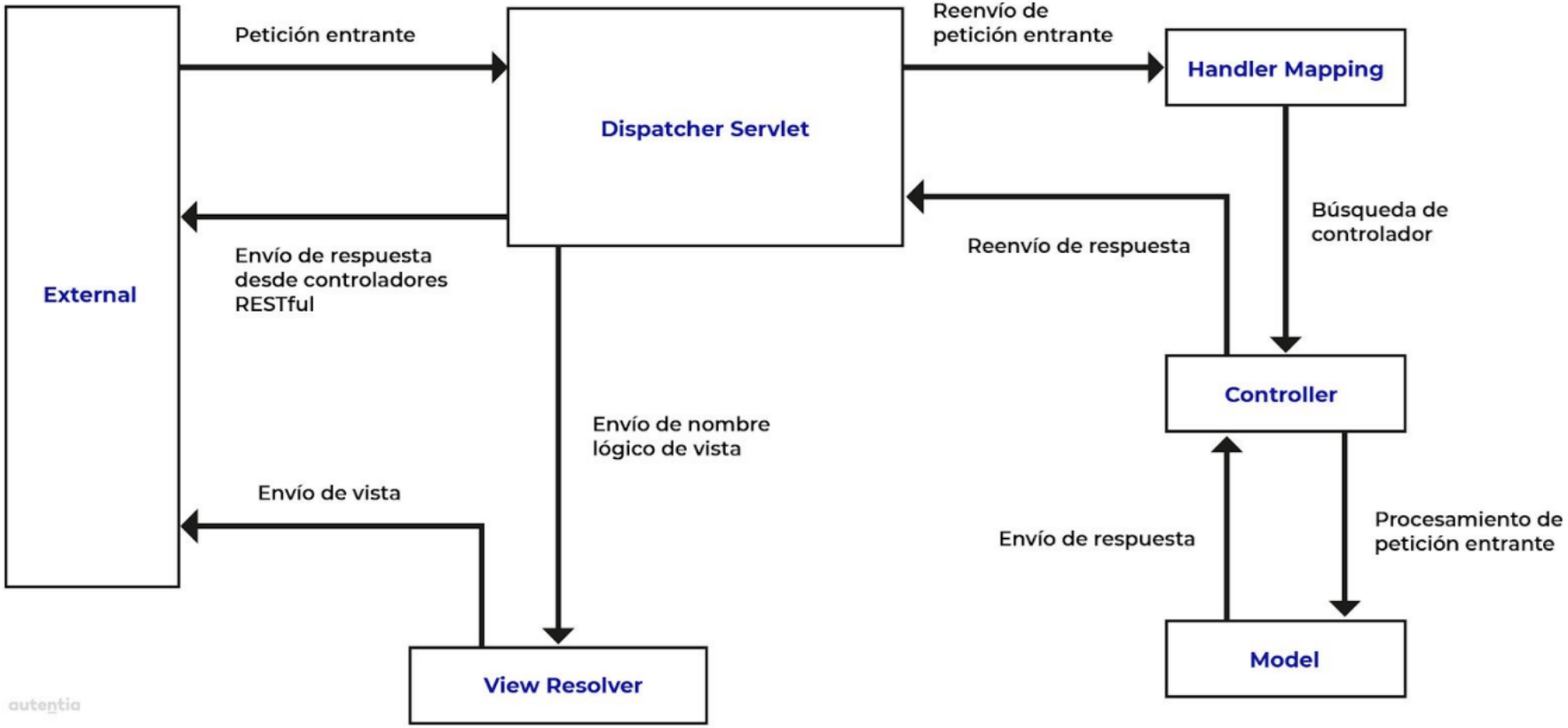
```
public class User {

    @Id
    private Long id;
    private String firstName;
    private String lastName;

    // constructors, getters, setters
}
```

Spring Data JDBC

```
@Repository
public interface UserRepository extends CrudRepository<User, Long>
{}
```

Spring MVC

- Framework basado en el **patrón MVC**
- **DispatcherServlet** procesa peticiones entrantes y redirige entre los distintos controladores
- **Modelo**: encapsula los datos de la aplicación
- **Vista**: representa los datos del modelo en un formato específico (HTML)
- **Controlador**: actúa de puente entre la vista y el modelo.


External
Petición entrante
Dispatcher Servlet
Reenvío de
petición entrante
Handler Mapping
Búsqueda de
controlador
Reenvío de respuesta
Envío de respuesta
desde controladores
RESTful
Controller
Envío de nombre
lógico de vista
Envío de vista
Procesamiento de
petición entrante
Envío de respuesta
Model
View Resolver

Spring MVC

```
@Controller
public class HomeController {

@RequestMapping(value ="/home", method = RequestMethod.GET)
public String home(Model model) {
    String msg = "Hello World";
    model.addAttribute("msgInView", msg);
    return "home";
}
}
```

```
<html>
<head>
   <title>Home</title>
</head>
<body>
  <p>${msgInView}</p>
</body>
</html>
```

Spring MVC - Rest

```
@RestController
public class ServiceController {
 @RequestMapping(value="/servicios", method= RequestMethod.GET)
 public List<Service> serviceListing() {
 // TODO
 }
}
```

Spring MVC - Thymeleaf

```
<div class="seedstarterlist" th:unless="${#lists.isEmpty(allSeedStarters)}">

  <h2 th:text="#{title.list}">List of Seed Starters</h2>

  <table>
    <thead>
      <tr>
        <th th:text="#{seedstarter.datePlanted}">Date Planted</th>
        <th th:text="#{seedstarter.covered}">Covered</th>
        <th th:text="#{seedstarter.type}">Type</th>
        <th th:text="#{seedstarter.features}">Features</th>
        <th th:text="#{seedstarter.rows}">Rows</th>
      </tr>
    </thead>
    <tbody>
      <tr th:each="sb : ${allSeedStarters}">
        <td th:text="${{sb.datePlanted}}">13/01/2011</td>
        <td th:text="#{|bool.${sb.covered}|}">yes</td>
        <td th:text="#{|seedstarter.type.${sb.type}|}">Wireframe</td>
        <td th:text="${#strings.arrayJoin(
                        #messages.arrayMsg(
                            #strings.arrayPrepend(sb.features,'seedstarter.feature.')),
                        ', ')}">Electric Heating, Turf</td>
        <td>
          <table>
            <tbody>
              <tr th:each="row,rowStat : ${sb.rows}">
                <td th:text="${rowStat.count}">1</td>
                <td th:text="${row.variety.name}">Thymus Thymi</td>
                <td th:text="${row.seedsPerCell}">12</td>
              </tr>
            </tbody>
          </table>
        </td>
      </tr>
    </tbody>
  </table>
</div>
```


NOW I WILL TAKE
YOUR QUESTIONS.

Trabaja con nosotros

LOS MEJORES EQUIPOS

Macbook Pro y cualquier material que necesites para trabajar.

SEGURO DE SALUD

Elige el profesional de la salud que necesites en cada ocasión.

HORARIO FLEXIBLE

Adapta el horario a tu vida personal entrando antes o después.

FORMACIÓN

Renueva tus conocimientos con charlas y cursos internos.

TICKETS RESTAURANT

Come donde quieras con este complemento económico diario.

FIDIOTERAPEUTA

Recupera tu condición física con la ayuda de un profesional.

Servicios

SOPORTE A DESARROLLO

Construimos entornos sólidos para los proyectos, trabajando a diario con los equipos de desarrollo.

DISEÑO DE PRODUCTO UX

Convertimos tus ideas en productos de valor para los usuarios finales.

ACOMPAÑAMIENTO AGILE

Implantamos, acompañamos y ayudamos a escalar modelos ágiles en las organizaciones.

SOFTWARE A MEDIDA

Estamos especializados en Aplicaciones web y desarrollos móviles e híbridos.

AUDITORIA DE DESARROLLO

Analizamos la calidad técnica de tu producto y te ayudamos a recuperar la productividad perdida.

FORMACIÓN

Cursos para empresas impartidos por profesionales en activo.

En qué creemos

COMPARTE TU CONOCIMIENTO

La enseñanza recíproca minimiza esfuerzos a la hora de aprender. Tú me enseñas y yo te enseño.

NO DEJES DE APRENDER

Somos curiosos, nuestras ganas de aprender son infinitas y por eso nunca paramos de formarnos.

NADA ES IMPOSIBLE

Nos encanta lo que hacemos y por eso nunca falta motivación e iniciativa para afrontar un reto.

DEMUESTRA LO QUE SABES HACER

No importan la edad ni los títulos académicos, sólo los conocimientos que tengas y lo que sepas hacer.

TRAE TORTILLA CON CEBOLLA

Crea un buen ambiente para trabajar en equipo, donde cualquier compañero esté dispuesto a echar una mano.

NO TODOS LOS CAMINOS LLEVAN A ROMA

Valoramos la calidad final, pero también la metodología, procesos y buenas prácticas en un proyecto.