
Curso de Hibernate y Java Spring

Marco Antonio Toscano



Marco Antonio Toscano

- Quito - Ecuador
- Ingeniero en Sistemas
Escuela Politécnica Nacional
- Experto en tecnologías Java, Linux, Open Source



www.youtube.com/user/matoosfe



[@martosfre](https://twitter.com/martosfre)

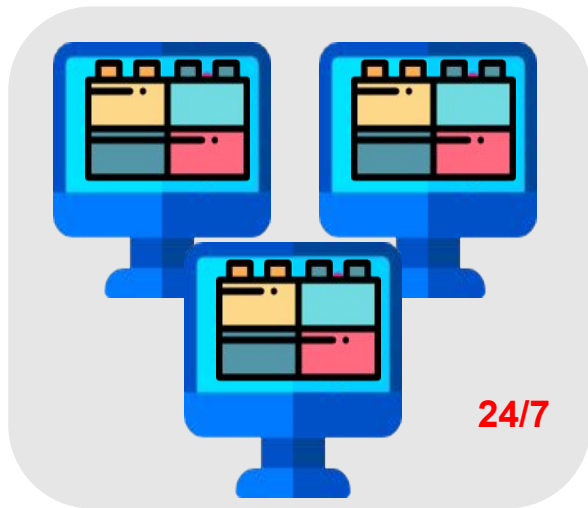


www.marcotoscano.org

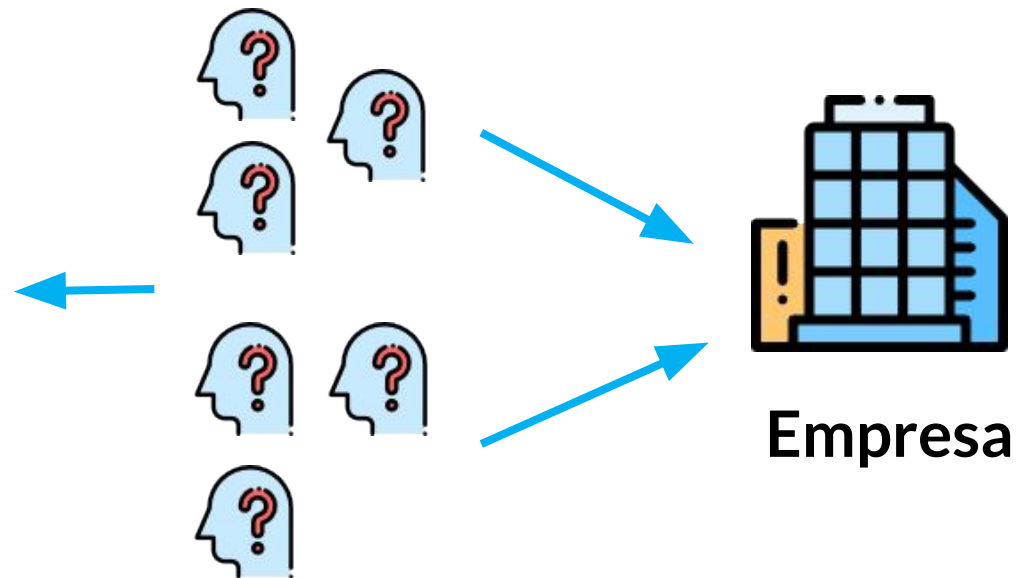
Aplicaciones Web

The bottom-left corner of the slide features decorative geometric shapes. It includes a dark blue triangle pointing towards the center, a semi-circle, and a square, all in varying shades of blue.

¿Qué es una aplicación empresarial?



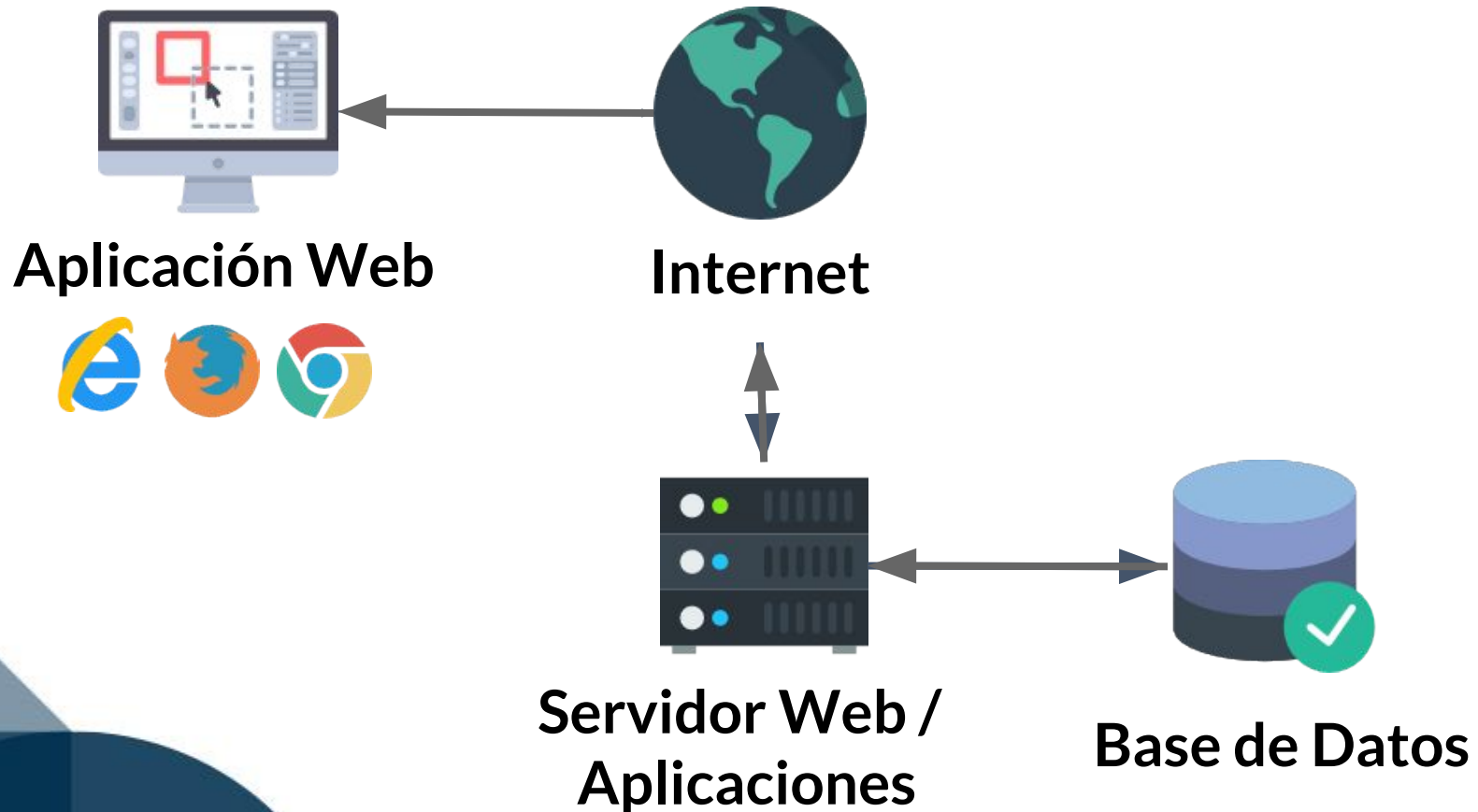
**Aplicación
Empresarial**



Necesidades

Empresa

¿Qué es una aplicación web?



¿Cómo acceder a una aplicación web?

Se realiza a través de una URL (dirección web)

http://192.168.1.170:8080/appventas

Protocolo

IP Servidor
Web/Aplicación

Puerto

Contexto
Aplicación

https://www.platzi.com/appventas

Protocolo

Dominio

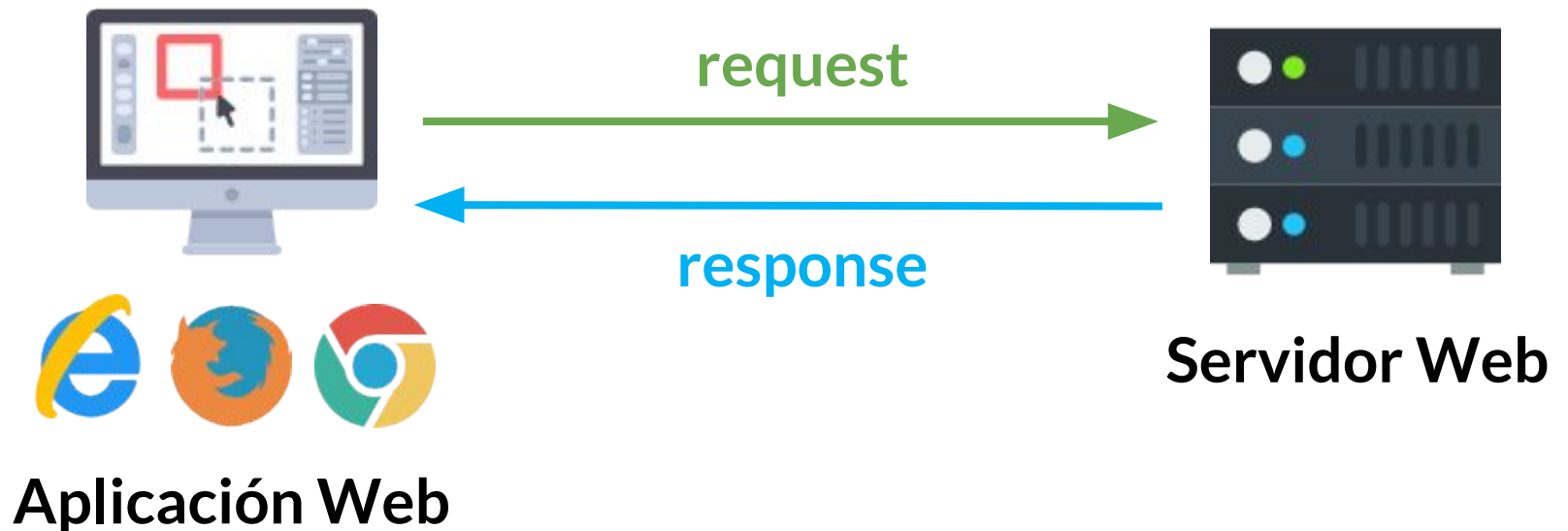
Contexto Aplicación

¿Qué es una aplicación web, ventajas?

- Fácil instalación y actualización.
- **Ahorro de recursos** en equipos y dispositivos.
- Independencia del Sistema Operativo -> **compatibilidad multiplataforma.**
- Soporte múltiples usuarios concurrentes.
- Acceso multidispositivo: computadora, tablet, tv, teléfono móvil.

¿Qué es un servidor web?

`http://192.168.1.145:8080/app`

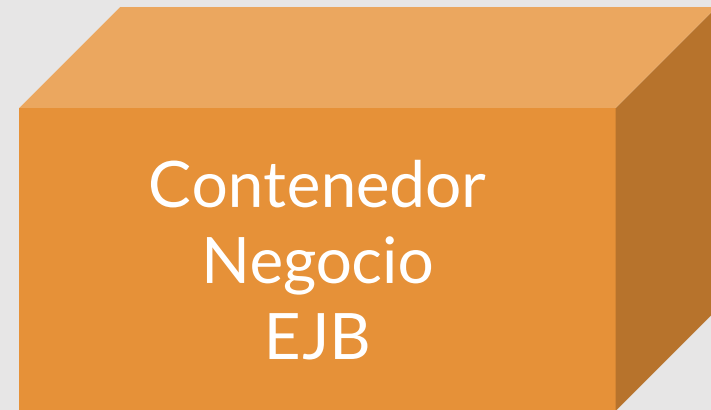
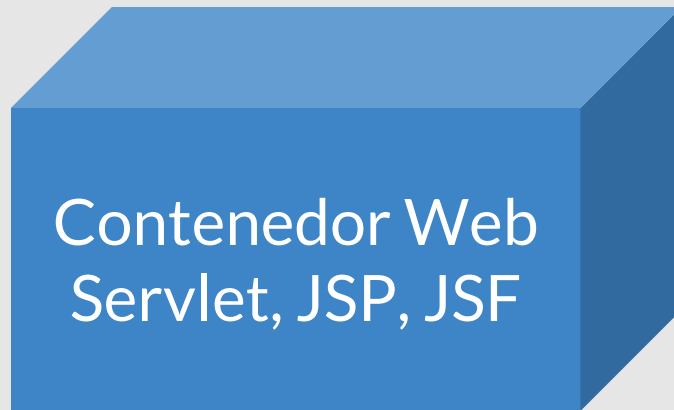


¿Qué es un servidor web?



Servidor Web

¿Qué es un servidor aplicación?



Servidor Aplicación

¿Cómo instalar ambiente de Desarrollo?

Para trabajar en el desarrollo de las aplicaciones necesitamos las siguientes herramientas:

- Java Developer Kit - JDK 1.8 o superior
- Maven 3.6
- Spring Tool Suite 4
- Postgresql 9.6 o superior (Docker)
- OmniDB

Instalar Maven Spring Tool Suite



Patrón de Diseño MVC

Modelo Vista Controlador
M.V.C

Modelo

Clases Mapeadas que representan tablas de la base de datos.

Negocio

Clases que representan el negocio propiamente.

Vista

Clases, archivos relacionados que representan la UI con las que interactúa el usuario.

¿Qué son los Servlets?

Son componentes del lado del servidor (clase java) que permiten **procesar peticiones del cliente** y responderlas a través de la generación de contenido dinámico o redireccionarlas a otros recursos.

¿Qué son los Servlets?

Hay dos tipos de Servlet

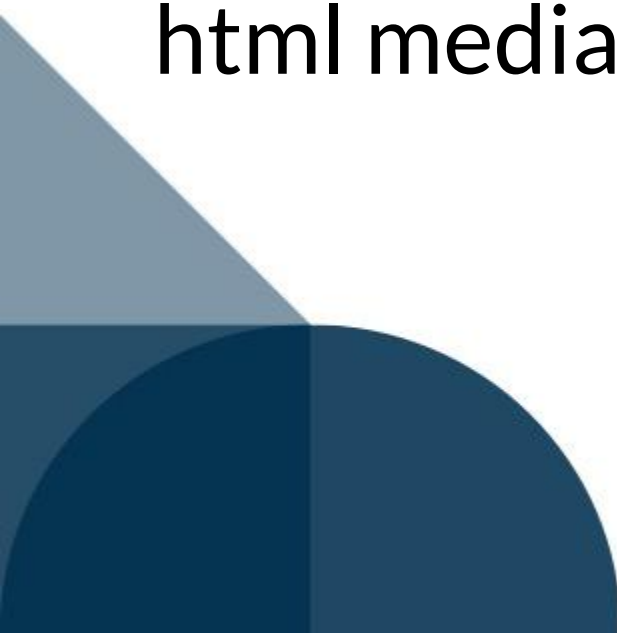
- GenericServlet (Otros Protocolos), las clases relacionadas se encuentran en el paquete **javax.servlet**
- HttpServlet(HTTP), las clases relacionadas se encuentran en el paquete **javax.servlet.http**.

Se pueden configurar a través de anotaciones o xml.

Demo Servlet

¿Qué son las JSP?

Son componentes del lado del servidor para desarrollar páginas web que soportan contenido dinámico gracias a la inclusión de código java en código html mediante el uso tags JSP.



Demo JSP

Introducción a Spring Boot

¿Qué es la arquitectura de la aplicación?



Requerimientos



Diseño Casa



Casa Terminada

PROCESO



Requerimientos



Diseño Aplicación



Aplicación
Terminada

¿Qué es Spring Framework?


- Framework Open Source. Rod Johnson 2003.
- Fue creado como una alternativa para solucionar la complejidad de las tecnologías Java más pesadas en ese momento, especialmente **EJB**.
- Spring no está limitado al lado del servidor.
- Nació como el core, actualmente es una empresa comercial con varios proyectos de diferente ámbito.

Características de Spring

- Desarrollo Basado en POJOS.
- Bajo acoplamiento - DI.
- Programación Declarativa.
- Reducción Boilerplate code.
- Arquitectura en Capas.

¿Qué es Spring Boot?

Es una tecnología de Spring que permite optimizar los tiempos de desarrollo en la **creación y despliegue de los proyectos**, permitiéndonos enfocar en el desarrollo de la aplicación.



Características Spring Boot

- Aplicaciones Spring standalone.
- Servidores web embebidos.
- Configuración simple.
- Características de producción listas.

¿Cómo crear una aplicación en Spring Boot?

- Requerimientos (JDK, Maven, IDE)
- Aplicación a Construir
- Tecnología a utilizar.
 - Spring Data JPA.
 - Spring Rest.
- Utilizar Spring **Initializr**



Crear la aplicación e-reservation

SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Java and Spring Boot 2.1.1

Project Metadata

Artifact coordinates

Group

com.platzzi

Artifact

e-reservation

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Web, Security, JPA, Actuator, Devtools...

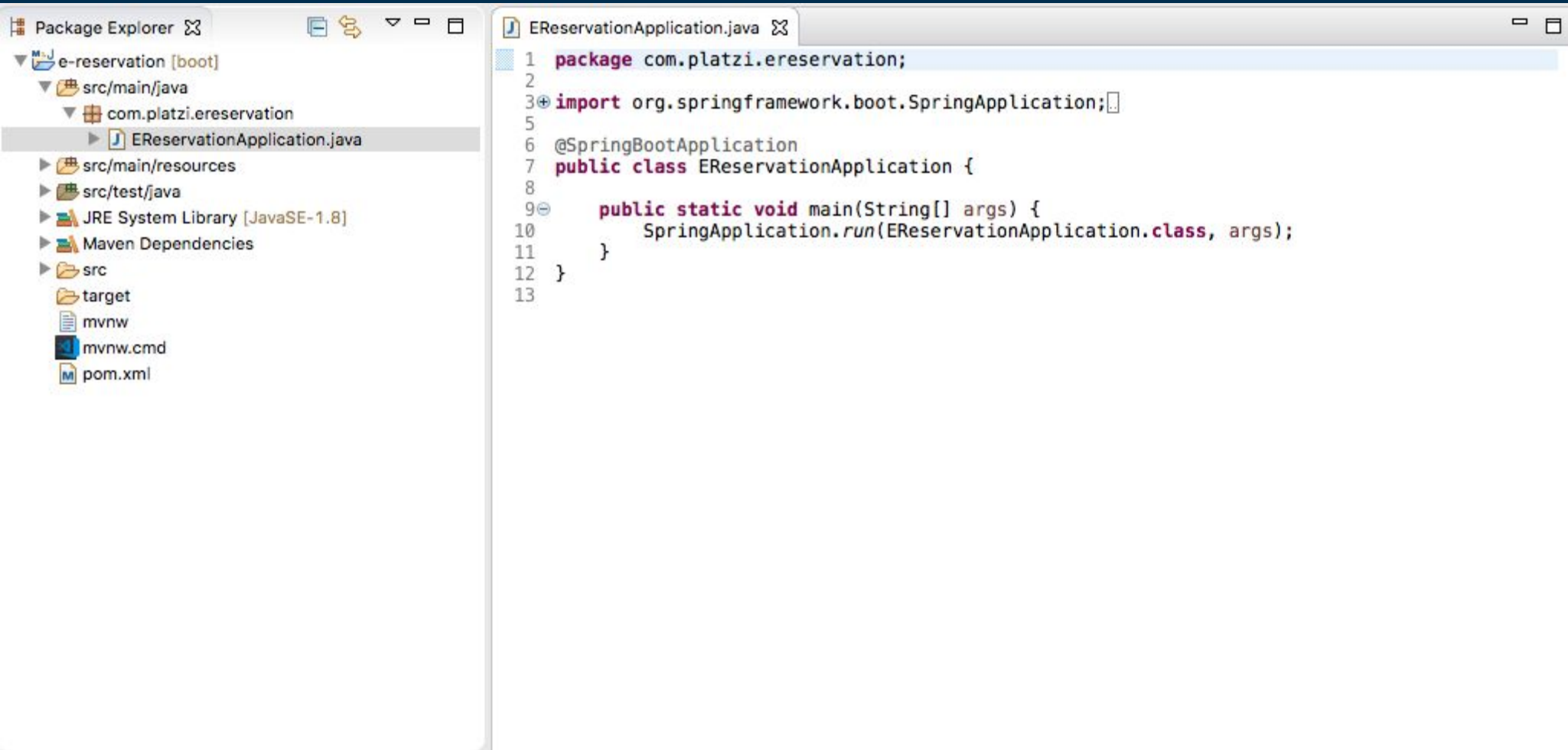
Selected Dependencies

Web X

Generate Project   

Don't know what to look for? Want more options? [Switch to the full version.](#)

Cargar y Configurar la aplicación



¿Cómo crear la estructura del Proyecto?

E-reservation.jar



Modelo Vista Controlador M.V.C

PACKAGE BASE
`com.platzi.ereservation`

Modelo

`ereservation.modelo`
`Cliente.java`
`Reserva.java`

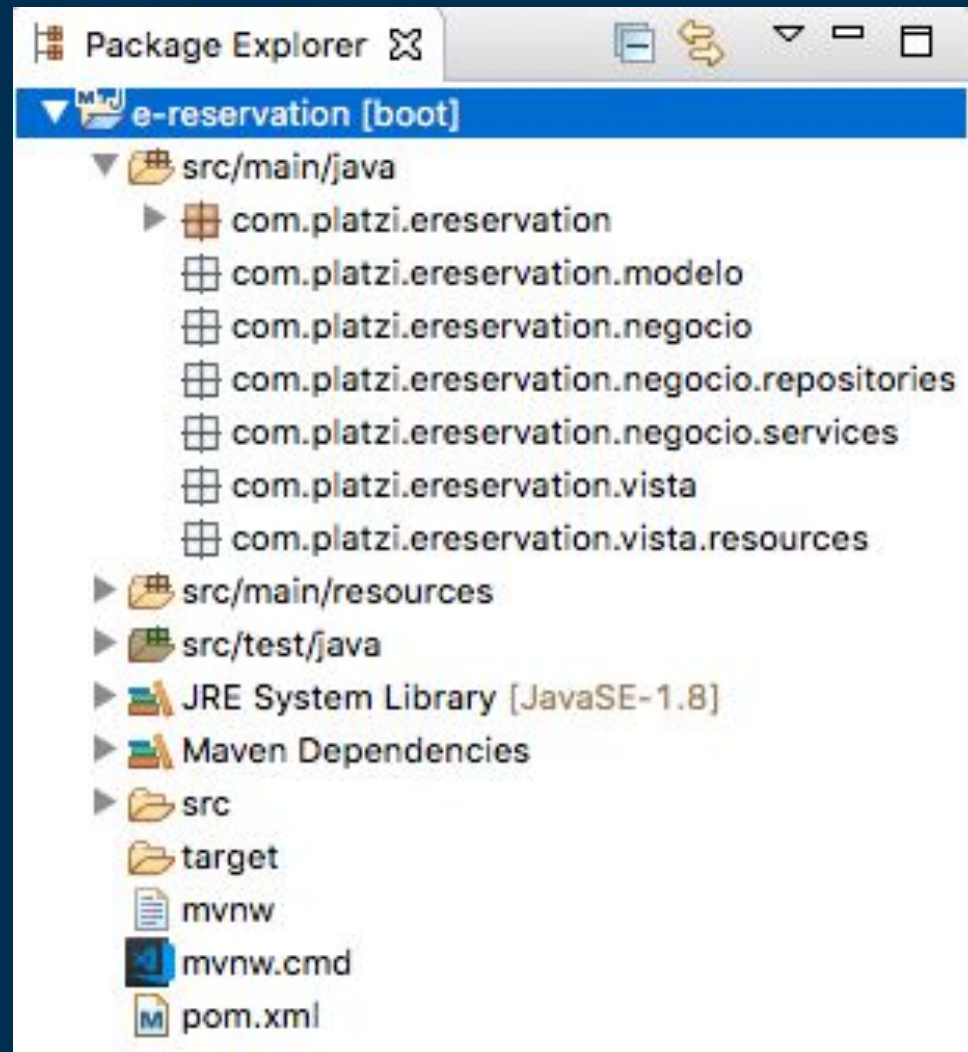
Negocio

`ereservation.repository`
`ClienteRepository.java`
`ereservation.services`
`ClienteService.java`

Vista

`ereservation.resources`
`ClienteResource.java`

Crear la estructura del Proyecto



Spring Data JPA



¿A qué se le conoce como persistencia?

- Almacenar información de tal manera de poder recuperarla en algún momento íntegra tal como guardó.
- Se tiene varios mecanismos de persistencia: archivos planos, archivos binarios, base de datos relacional, documental, etc.
- JDBC-> ORM(Hibernate, EclipseLink..) -> JPA

¿Qué es Spring Data?

- Manejar datos de las aplicaciones con un modelo de programación basado en Spring.
- Soporta diferentes tecnologías de base de datos tanto relacionales como no relacionales, frameworks map-reduce y servicios de datos basado en la nube.

¿Qué es Spring Data JPA?

- Provee una manera fácil de implementar la capa de acceso a datos utilizando la especificación de JPA.
- Spring Data JPA creará una implementación de **repositorios** (una capa más alta) y encapsulará todos los detalles de una implementación de JPA.
- Soporte paginación, ejecución dinámica de queries.

Inicializar la base de datos Postgres utilizando Docker y OmniDB



PostgreSQL



OMNIDB

Inicializar la base de datos Postgres utilizando Docker y OmniDB

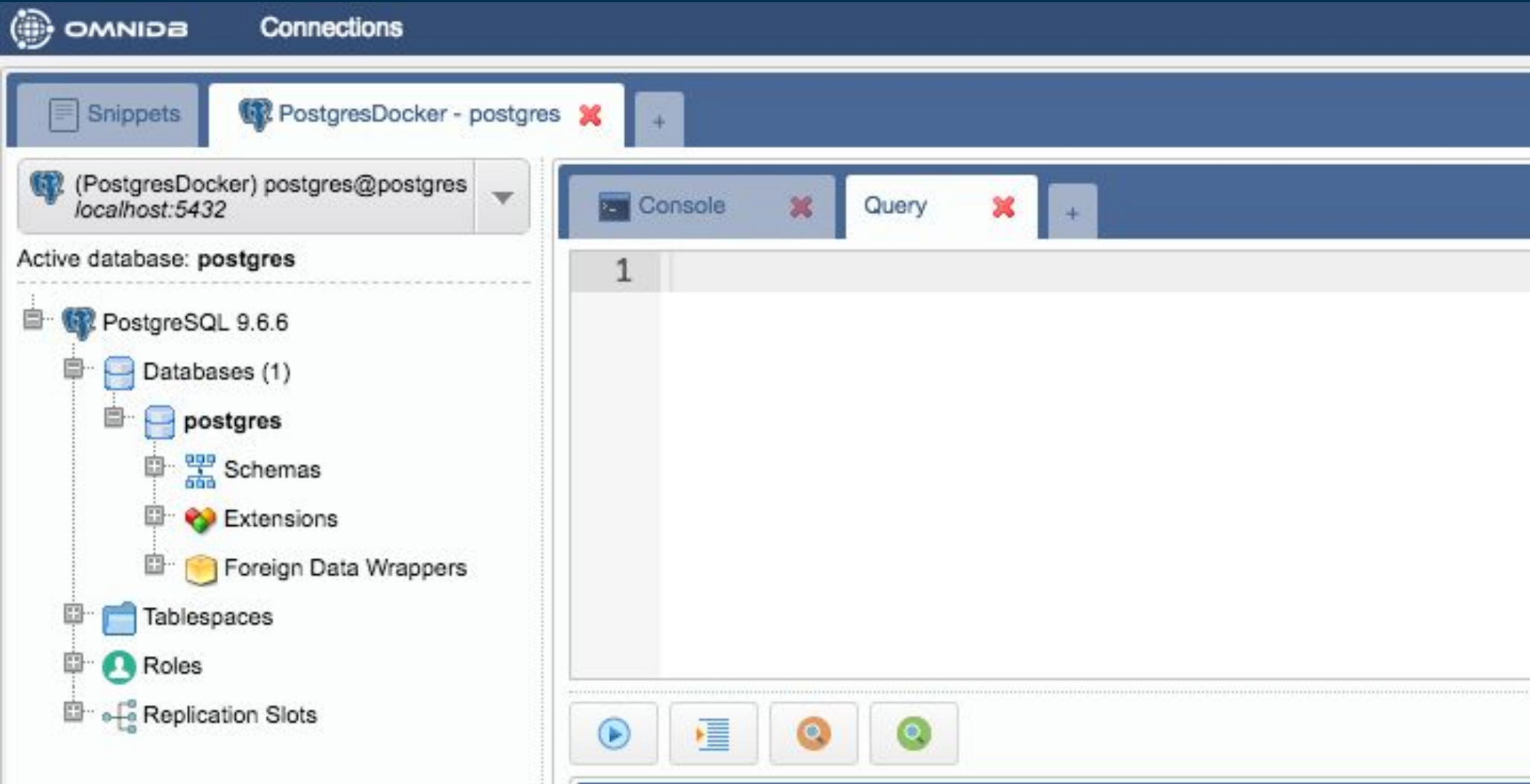
```
MacBook-Pro-de-Marco:~ martosfre$ docker pull postgres:9.6.6-alpine
9.6.6-alpine: Pulling from library/postgres
Digest: sha256:bf87ee22821e1bc5cedd5da2def1700685a9e3828605b31162d8f04e16c06385
Status: Image is up to date for postgres:9.6.6-alpine
MacBook-Pro-de-Marco:~ martosfre$
```

```
MacBook-Pro-de-Marco:~ martosfre$ docker run -d --name postgres -p 5432:5432 -e POSTGRES_PASSWORD=platzi
postgres:9.6.6-alpine
0ba4381c0f2488c7af1881742f50a8a8f043732ca04cf6d246cba17370e565df
MacBook-Pro-de-Marco:~ martosfre$ docker ps -a
```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS |
|------------------------|-----------------------|--------------------------|---------------|--------------|
| 0ba4381c0f24 | postgres:9.6.6-alpine | "docker-entrypoint.s..." | 5 seconds ago | Up 3 seconds |
| 0.0.0.0:5432->5432/tcp | postgres | | | |

```
MacBook-Pro-de-Marco:~ martosfre$
```

Inicializar la base de datos Postgres utilizando Docker y OmniDB



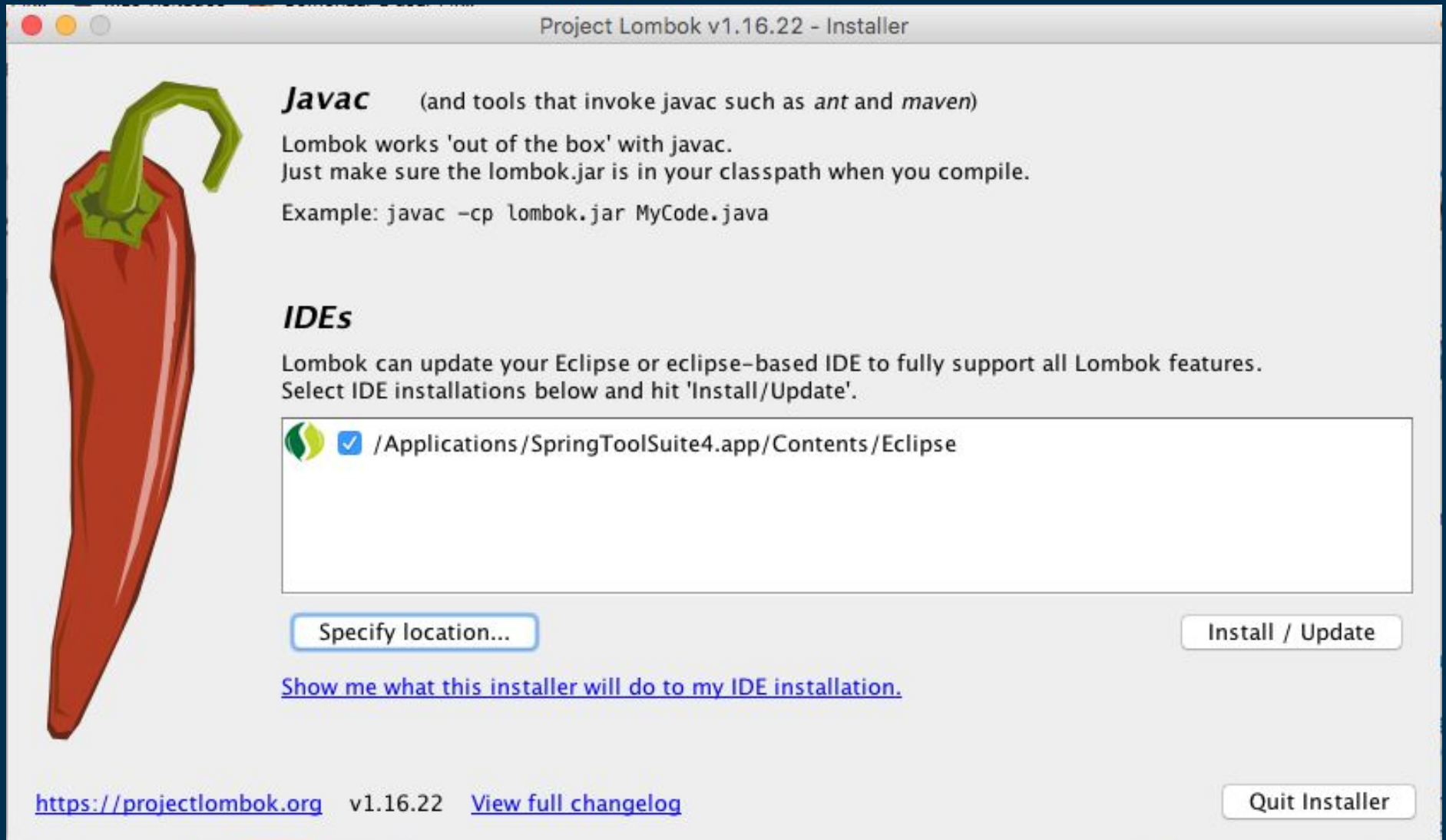
¿Qué es Lombok?



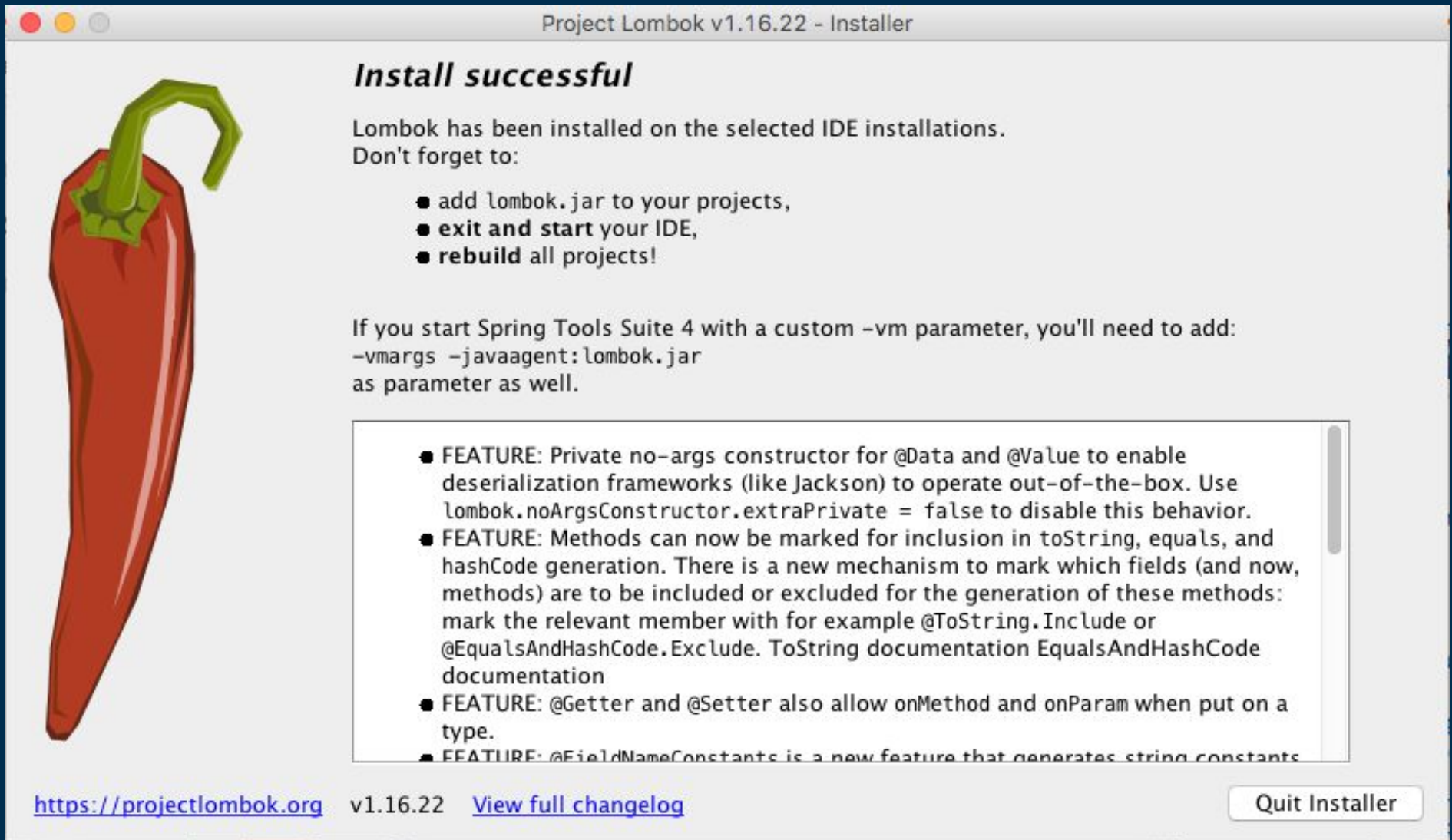
Proyecto
Lombok

- Es una librería que nos ayuda a eliminar código repetitivo en nuestras aplicaciones Java haciéndonos más productivos a través del uso de anotaciones.
- Se integra con el IDE.

¿Cómo instalar Lombok en el IDE?



¿Cómo instalar Lombok en el IDE?



¿Cómo instalar Lombok en la aplicación?

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.16.16</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```


Configurar proyecto para el Mapeo

```
<!-- Dependencias de JPA -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <version>42.1.4</version>
</dependency>
```

¿Cómo realizar el mapeo de clases?

table cliente

| idCli | | | |
|-------|--|--|--|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

```
@Data
@Entity
@Table(name = "cliente")
public class Cliente {
    @Id
    @GeneratedValue(generator="system-uuid")
    @GenericGenerator(name="system-uuid", strategy="uuid2")
    private String idCli;
    private String nombreCli;
    private String apellidoCli;
    private String identificacionCli;
    private String direccionCli;
    private String telefonoCli;
}
```

¿Cómo realizar el mapeo de clases?

```
@Data
@Entity
@Table(name = "cliente")
public class Cliente {
    @Id
    @GeneratedValue(generator="system-uuid")
    @GenericGenerator(name="system-uuid", strategy="uuid2")
    private String idCli;
    private String nombreCli;
    private String apellidoCli;
    private String identificacionCli;
    private String direccionCli;
    private String telefonoCli;
    private String emailCli;

    @OneToMany(mappedBy="cliente")
    private Set<Reserva> reservas;

    public Cliente() {
    }
}
```

¿Cómo realizar el mapeo de clases?

```
@Data
@Entity
@Table(name="reserva")
public class Reserva {
    @Id
    @GeneratedValue(generator="system-uuid")
    @GenericGenerator(name="system-uuid", strategy="uuid2")
    private String idRes;
    private String codigoRes;
    @Temporal(TemporalType.DATE)
    private Date fechaIngresoRes;
    @Temporal(TemporalType.DATE)
    private Date fechaSalidaRes;
    private int cantidadPersonasRes;
    private String descripcionRes;
    @ManyToOne
    @JoinColumn(name="idCli")
    private Cliente cliente;
}
```


¿Cómo configurar persistencia en Hibernate?

```
1 spring:
2   datasource:
3     url: jdbc:postgresql://localhost:5432/postgres
4     username: postgres
5     password: platzi
6     driver-class-name: org.postgresql.Driver
7   jpa:
8     hibernate:
9       ddl-auto: create-drop
10      generate-ddl: true
11      show-sql: true
```

Manejo del Negocio

The bottom left corner of the slide features decorative geometric shapes. It includes a dark blue triangle pointing towards the center and a semi-circle in a slightly lighter shade of blue, partially overlapping the triangle.

¿Cómo configurar un JPA Repository en nuestra aplicación?

- La herramienta principal de Spring Data son los repositorios. El repositorio principal o base es Repository, de la cual se tiene interfaces centrales CrudRepository, PageAndSortingRepository.
- Existen varios repositorios de acuerdo a la tecnología a utilizar como JpaRepository, MongoRepository que se basan en PageAndSortingRepository

```
/**
 * Clase para manejar las operaciones de negocio relacionadas con Cliente.
 * @author martosfre
 */
public interface ClienteRepository extends JpaRepository<Cliente, String> {
}
```

```
/**
 * Clase para manejar las operaciones de negocio relacionadas con la Reserva.
 * @author martosfre
 */
public interface ReservaRepository extends JpaRepository<Reserva, String> {
}
```

¿Cómo implementar las operaciones de consulta en los repositorios?

- Spring Data cuenta con soporte para realizar consultas personalizadas basadas en los atributos de la clase y que éstas se generen de forma automática en tiempo de ejecución (**query generation strategy**)
- También soporta otro tipo de consultas basadas en JPQL como son @NamedQuery y @Query.

```
public interface ClienteRepository extends JpaRepository<Cliente, String> {  
  
    /**  
    * Método para buscar un cliente por su identificación  
    * @param identificacionCli  
    * @return  
    */  
    public Cliente findByIdentificacionCli(String identificacionCli);  
  
    /**  
    * Método para buscar los clientes por el nombre y apellido  
    * @param nombreCli  
    * @param apellidoCli  
    * @return  
    */  
    public List<Cliente> findByNombreCliAndApellidoCli(String nombreCli, String apellidoCli);  
}
```

¿Cómo implementar un servicio de negocio con las operaciones CRUD?

- Una vez que se tenga definido nuestro repositorio, el siguiente paso es trabajar en las clases para exponer estas operaciones en la capa de negocio las cuales son llamados como servicios.
- Para indicar que la clase es un bean de la capa de negocio se utiliza la anotación **@Service**.

```

1  /**
2   * Clase para manejar los servicios de cliente
3   * @author martosfre
4   */
5  public class ClienteService {
6
7      private final ClienteRepository clienteRepository;
8
9      * @param clienteRepository
10     public ClienteService(ClienteRepository clienteRepository) {
11         this.clienteRepository = clienteRepository;
12     }
13
14     * Método para crear un cliente
15     public Cliente create(Cliente cliente) {
16         return this.clienteRepository.save(cliente);
17     }
18
19     * Método para actualizar un cliente
20     public Cliente update(Cliente cliente){
21         return this.clienteRepository.save(cliente);
22     }
23
24     * Método para eliminar un cliente
25     public void delete(Cliente cliente) {
26         this.clienteRepository.delete(cliente);
27     }
28
29     * Método para buscar todos los clientes
30     public List<Cliente> findAll(){
31         return this.clienteRepository.findAll();
32     }
33 }

```

¿Cómo manejar el tema transaccional en una operación de negocio?

- Transacción es el conjunto de operaciones que se ejecutan en bloque, todas o ninguna.
- Las transacciones empiezan y terminan a nivel de servicio nunca a nivel de capa de datos.
- Para indicar que una clase o método será transaccional se utiliza la anotación **@Transactional**

```

@Transactional(readOnly = true)
@Service
public class ClienteService {

    private final ClienteRepository clienteRepository;

    + * @param clienteRepository[]
    - public ClienteService(ClienteRepository clienteRepository) {
        this.clienteRepository = clienteRepository;
    }

    + * Método para crear un cliente[]
    - @Transactional
    public Cliente create(Cliente cliente) {
        return this.clienteRepository.save(cliente);
    }

    + * Método para actualizar un cliente[]
    - @Transactional
    public Cliente update(Cliente cliente) {
        return this.clienteRepository.save(cliente);
    }

    + * Método para eliminar un cliente[]
    - @Transactional
    public void delete(Cliente cliente) {
        this.clienteRepository.delete(cliente);
    }

    + * Método para buscar todos los clientes[]
    - public List<Cliente> findAll() {
        return this.clienteRepository.findAll();
    }

```

Spring Rest

The bottom-left corner of the slide features a decorative design consisting of a dark blue triangle and a semi-circle, both in a slightly darker shade than the background.

¿Qué son los servicios web?

- Son aplicaciones construidas para poder intercambiar información entre otras aplicaciones utilizando protocolos estandarizados - **interoperabilidad**.
- Existen dos maneras de construir servicios web: SOAP y **REST**



Características de Spring Rest

- Es una manera de construir servicios web en Spring utilizando la arquitectura REST aprovechando la experiencia de Spring MVC.
- Para configurar un servicios web se utiliza la anotación **@RestController** en la clase base, la cual combina los comportamientos de las anotaciones @Controller y @ResponseBody.

¿Cómo implementar las operaciones crud en REST?

```
@RestController
@RequestMapping("/api/cliente")
public class ClienteResource {

    private final ClienteService clienteService;

    public ClienteResource(ClienteService clienteService) {
        this.clienteService = clienteService;
    }

    @PostMapping
    public ResponseEntity<Cliente> createCliente(@RequestBody ClienteVO clienteVo) {
        Cliente cliente = new Cliente();
        cliente.setNombreCli(clienteVo.getNombreCli());
        cliente.setApellidoCli(clienteVo.getApellidoCli());
        cliente.setIdentificacionCli(clienteVo.getIdentificacionCli());
        cliente.setDireccionCli(clienteVo.getDireccionCli());
        cliente.setTelefonoCli(clienteVo.getTelefonoCli());
        cliente.setEmailCli(clienteVo.getEmailCli());
        return new ResponseEntity<>(this.clienteService.create(cliente), HttpStatus.CREATED);
    }
}
```

¿Cómo instalar y configurar Swagger?

- Es la herramienta por defecto para documentar API's Web, genera una forma interactiva de ejecutarlos.
- Provee una manera fácil de implementar clientes en variedad de lenguajes.
- Para configurarlo en Spring se utilizará el proyecto **Spring Fox** con sus dependencias en el archivo pom.xml del proyecto.

¿Cómo instalar y configurar Swagger?

- Para poder anotar con Swagger a los recursos REST se tiene primero que crear una clase de configuración y hacer que el contenedor de Spring lo reconozca para generar un bean a través de la anotación **@Configuration**
- Adicionalmente se tiene que habilitar Swagger con la anotación **@EnableSwagger2**.

```
<!-- Dependencias para Swagger -->
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>2.7.0</version>
</dependency>
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger-ui</artifactId>
  <version>2.7.0</version>
</dependency>
```

```
3 @Configuration
4 @EnableSwagger2
5 public class SwaggerConfiguration {
6     @Bean
7     public Docket documentation() {
8         return new Docket(DocumentationType.SWAGGER_2).select()
9             .apis(RequestHandlerSelectors.withClassAnnotation(RestController.class))
10            .paths(PathSelectors.any())
11            .build();
12     }
13 }
14 }
```


¿Cómo anotar los REST Resources con Swagger?

```
@RestController
@RequestMapping("/api/cliente")
@Api(tags = "cliente")
public class ClienteResource {

    private final ClienteService clienteService;


    public ClienteResource(ClienteService clienteService) {
        this.clienteService = clienteService;
    }

    @PostMapping
    @ApiOperation(value = "Crear Cliente", notes = "Servicio para crear un nuevo cliente")
    @ApiResponses(value = { @ApiResponse(code = 201, message = "Cliente creado correctamente"),
        @ApiResponse(code = 400, message = "Solicitud Inválida") })
    public ResponseEntity<Cliente> createCliente(@RequestBody ClienteVO clienteVo) {
        Cliente cliente = new Cliente();
        cliente.setNombreCli(clienteVo.getNombreCli());
        cliente.setApellidoCli(clienteVo.getApellidoCli());
        cliente.setIdentificacionCli(clienteVo.getIdentificacionCli());
        cliente.setDireccionCli(clienteVo.getDireccionCli());
        cliente.setTelefonoCli(clienteVo.getTelefonoCli());
        cliente.setEmailCli(clienteVo.getEmailCli());
        return new ResponseEntity<>(this.clienteService.create(cliente), HttpStatus.CREATED);
    }
}
```

¿Cómo anotar los REST Resources con Swagger?

localhost:8080/swagger-ui.html

Comenzar a usar Fir... Más visitados Comenzar a usar Fir...

 **swagger** default (/v2/api-docs) **Explore**

Api Documentation

Api Documentation

[Apache 2.0](#)

cliente : Cliente Resource Show/Hide | List Operations | Expand Operations

| | | |
|---------------|-------------------------------|--------------------|
| GET | /api/cliente | Lista de clientes |
| POST | /api/cliente | Crear Cliente |
| DELETE | /api/cliente/{identificacion} | Eliminar cliente |
| PUT | /api/cliente/{identificacion} | Actualizar cliente |

[BASE URL: / , API VERSION: 1.0]

Spring Security

The bottom-left corner of the slide features decorative geometric shapes. It includes a dark blue triangle pointing towards the top-right, a medium blue square, and a large dark blue semi-circle.


¿Cómo manejar la seguridad en una aplicación web?

Mecanismos relacionados con la autenticación y autorización como son:

- Filtros de Autenticación
- Control Páginas de Error
- Envío de peticiones seguras

¿Cómo manejar la seguridad en una aplicación web?

Para controlar los problemas asociados a estos temas se tiene varios frameworks como son **Spring Security**, **Apache Shiro** entre otros que nos ayuden con estas tareas.



¿Qué es Spring Security?

Es un **framework de Spring** que permite gestionar completamente la seguridad de nuestras aplicaciones Java. Permite:

- Gestionar seguridad en varios niveles.
- Configuración de seguridad portable.
- Soporta muchos modelos de autenticación (HTTP Basic, LDAP, OAuth, Http Digest entre otros).

¿Cómo instalar y configurar Spring Security?

- Configurar la dependencia security-starter.
- Excluir la configuración por defecto y crear nuestra configuración de Spring Security.
- Configurar las credenciales del usuario en el archivo application.properties.
Por defecto viene configurado un usuario.

¿Cómo instalar y configurar Spring Security?

```
<!-- Dependencias Spring Security -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

```
package com.platzi.ereservation;

+ import org.springframework.boot.SpringApplication;

@SpringBootApplication(exclude = { SecurityAutoConfiguration.class })
public class EReservationApplication {

-     public static void main(String[] args) {
        SpringApplication.run(EReservationApplication.class, args);
    }
}
```

¿Cómo instalar y configurar Spring Security?

```
/**
 * Clase para manejar el tema de seguridad
 *
 * @author martosfre
 *
 */
@Configuration
@EnableWebSecurity
public class BasicConfiguration extends WebSecurityConfigurerAdapter {
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .anyRequest().authenticated()
            .and()
            .formLogin()
            .and()
            .httpBasic();
    }
}
```

Despliegue de aplicaciones Docker

The bottom-left corner of the slide features decorative geometric shapes. It includes a dark blue triangle pointing towards the center and a semi-circle with a dark blue gradient, also pointing towards the center.

¿Cómo se realiza la configuración de Docker Plugin?

Una vez terminada nuestra aplicación vamos a proceder a dockerizarla:

- Tener instalado docker en nuestro equipo.
- Configurar el plugin de docker en nuestro pom.xml.

```
<plugin>
  <groupId>io.fabric8</groupId>
  <artifactId>docker-maven-plugin</artifactId>
  <version>0.21.0</version>
  <configuration>
    <images>
      <image>
        <name>martosfre/${project.build.finalName}</name>
        <build>
          <from>openjdk:latest</from>
          <entryPoint>java -Dspring.profiles.active=prod -jar /Users/martosfre/application/${project.build.finalName}.jar</entryPoint>
          <assembly>
            <basedir>/Users/martosfre/application/</basedir>
            <descriptorRef>artifact</descriptorRef>
            <inline>
              <id>assembly</id>
              <files>
                <file>
                  <source>target/${project.build.finalName}.jar</source>
                </file>
              </files>
            </inline>
          </assembly>
          <tags>
            <tag>latest</tag>
          </tags>
          <ports>
            <port>8080</port>
          </ports>
        </build>
        <run>
          <namingStrategy>alias</namingStrategy>
        </run>
        <alias>${project.build.finalName}</alias>
      </image>
    </images>
  </configuration>
</plugin>
```

¿Cómo crear una imagen docker de la aplicación?

Después de configurar el plugin para construir una imagen de la aplicación se realiza lo siguiente:

- Configurar el nombre del proyecto final en el pom.xml.
- Crear una tarea de ejecución en el IDE para ejecutar el plugin y proceder a ejecutarla.

Una vez realizado se tendrá creada la imagen de nuestro proyecto y se podrá revisar con el comando **docker images**.

```
<build>
  <finalName>ereservation</finalName>
  <plugins>
```

Name: e-reservation_image

Main JRE Refresh Source Environment Common

Base directory: \${project_loc:e-reservation}

Workspace... File System... Variables...

Goals: clean install docker:build

Profiles:

User settings: /Users/martosfre/.m2/settings.xml

Workspace... File System... Variables...

☐ Offline ☐ Update Snapshots

☐ Debug Output ☐ Skip Tests ☐ Non-recursive

☐ Resolve Workspace artifacts

Revert Apply

? Close Run

Problems Javadoc Declaration Console Progress

<terminated> e-reservation_image [Maven Build] /Library/Java/JavaVirtualMachines/adoptopenjdk-8.jdk/Contents/Home/bin

[INFO] --- maven-jar-plugin:3.1.0:jar (default-jar) @ e-reservation ---

[INFO] Building jar: /Users/martosfre/Matoosfe/Proyectos/PlatziSpringBoot/Fuentes/e-reservation.jar

[INFO] --- spring-boot-maven-plugin:2.1.1.RELEASE:repackage (repackage) @ e-reservation ---

[INFO] Replacing main artifact with repackaged archive

[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ e-reservation ---

[INFO] Installing /Users/martosfre/Matoosfe/Proyectos/PlatziSpringBoot/Fuentes/e-reservation.jar to /Users/martosfre/.m2/repository/com/martosfre/e-reservation/1.0.0/e-reservation-1.0.0.jar

[INFO] Installing /Users/martosfre/Matoosfe/Proyectos/PlatziSpringBoot/Fuentes/e-reservation.jar to /Users/martosfre/.m2/repository/com/martosfre/e-reservation/1.0.0/e-reservation-1.0.0.jar

[INFO] --- docker-maven-plugin:0.21.0:build (default-cli) @ e-reservation ---

[INFO] Copying files to /Users/martosfre/Matoosfe/Proyectos/PlatziSpringBoot/Fuentes/e-reservation

[INFO] Building tar: /Users/martosfre/Matoosfe/Proyectos/PlatziSpringBoot/Fuentes/e-reservation.tar

[INFO] DOCKER> [martosfre/e-reservation:latest] "e-reservation": Created docker-build.tar

[INFO] DOCKER> [martosfre/e-reservation:latest] "e-reservation": Built image sha256:f2c5e

[INFO] DOCKER> [martosfre/e-reservation:latest] "e-reservation": Tag with latest

[INFO] -----






























[INFO] BUILD SUCCESS

[INFO] -----

[INFO] Total time: 33.125 s

[INFO] Finished at: 2018-12-07T16:55:17-05:00

[INFO] -----

- ▼  e-reservation [boot]
 - ▶  src/main/java
 - ▶  src/main/resources
 - ▶  src/test/java
 - ▶  JRE System Library [JavaSE-1.8]
 - ▶  Maven Dependencies
 - ▶  target/generated-sources/annotations
 - ▶  target/generated-test-sources/test-annotations
 - ▶  src
 - ▼  target
 - ▼  docker
 - ▼  martosfre
 - ▼  e-reservation
 - ▼  build
 - ▶  maven
 -  Dockerfile
 - ▶  tmp
 - ▶  work
 - ▶  build.timestamp
 - ▶  generated-sources
 - ▶  generated-test-sources
 - ▶  maven-archiver
 - ▶  maven-status
 - ▶  surefire-reports
-  e-reservation.jar
-  e-reservation.jar.original
- ▶  mvnw
- ▶  mvnw.cmd
- ▶  pom.xml

¿Cómo subir la aplicación a Docker Hub?

DockerHub es un repositorio público gratuito para almacenar imágenes docker.

- Logearse en Docker Hub desde la consola o terminal a través del comando `docker login`.
- Ejecutar el comando `docker push <image>:tag`

~/Matoosfe/Proyectos/PlatziSpringBoot/Fuentes/e-reservation — -bash

MacBook-Pro-de-Marco:e-reservation martosfre\$ docker images

| REPOSITORY | TAG | IMAGE ID | CREATED | SIZE |
|-------------------------|--------------|--------------|--------------------|--------|
| martosfre/e-reservation | latest | f2c5e13acfe3 | About a minute ago | 1.03GB |
| openjdk | latest | 8e7eacedab93 | 2 days ago | 986MB |
| postgres | 9.6.6-alpine | 7470b931fc2e | 11 months ago | 37.8MB |

MacBook-Pro-de-Marco:e-reservation martosfre\$ docker login

Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to [s://hub.docker.com](https://hub.docker.com) to create one.

Username: martosfre

Password:

Login Succeeded

MacBook-Pro-de-Marco:e-reservation martosfre\$ docker push martosfre/e-reservation:latest

The push refers to repository [docker.io/martosfre/e-reservation]

12a8f7dd0145: Pushed

fedd44c4ee44: Pushed

f762b1b7d799: Pushed

c9a8a600ae18: Pushed

eec9922d69ca: Pushed

c594d901dc9d: Pushed


b250144690ff: Pushed

d31e0e554b0a: Pushed

7034f4f565c8: Pushed

latest: digest: sha256:1676c964d9ba9b0f0b92bd5b51f12b09d7f3bdaf1c31ec77476588ec46dba8dc size: 2212

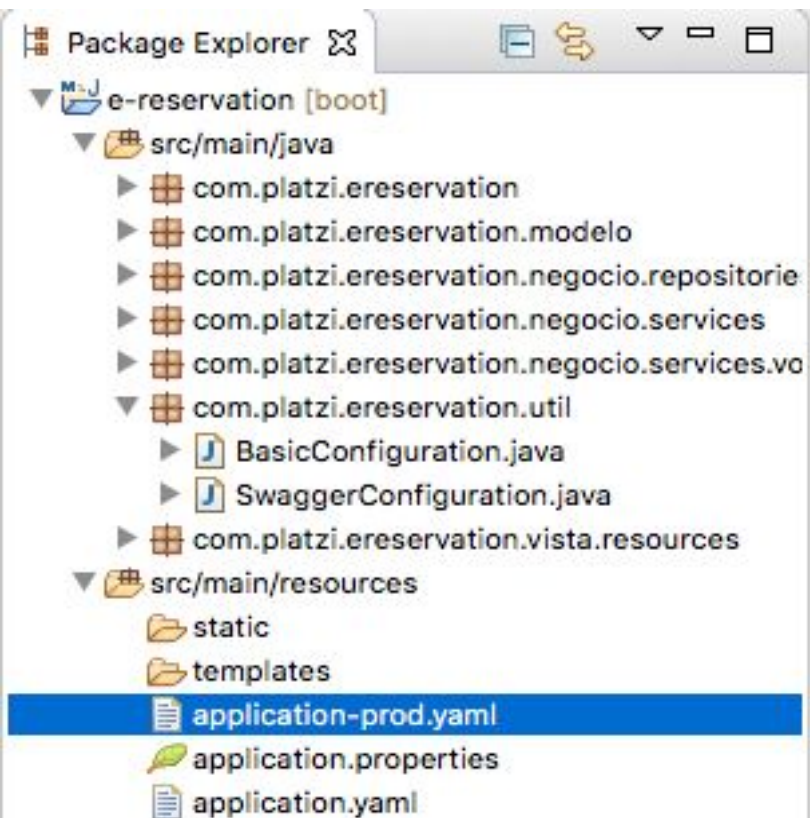
MacBook-Pro-de-Marco:e-reservation martosfre\$ █

| Repositories | | | | Create Repository + |
|---|--|--|--|---|
| Type to filter repositories by name | | | | |
| <div> martosfre/e-reservation public</div> | | | | <div>0 STARS</div> <div>0 PULLS</div> <div>> DETAILS</div> |

¿Cómo ejecutar la aplicación dockerizada?

Para ejecutar la aplicación dockerizada se tiene que realizar lo siguiente:

- Generar un spring-profile para configurar un ambiente de producción.
- Ejecutar la aplicación con el comando docker run enviando la configuración del servidor.



application-prod.yaml

```
1 spring:
2   datasource:
3     url: jdbc:postgresql://postgres_server:5432/postgres
4     username: postgres
5     password: platzi
6     driver-class-name: org.postgresql.Driver
7   jpa:
8     hibernate:
9       ddl-auto: create-drop
10      generate-ddl: true
11      show-sql: true
```

usr — -bash — 115x23

MacBook-Pro-de-Marco:usr martosfre\$ docker run -d --name ereservation --add-host=postgres_server:200.115.89.173 -p 8080:8080 martosfre/e-reservation:latest

Gracias

