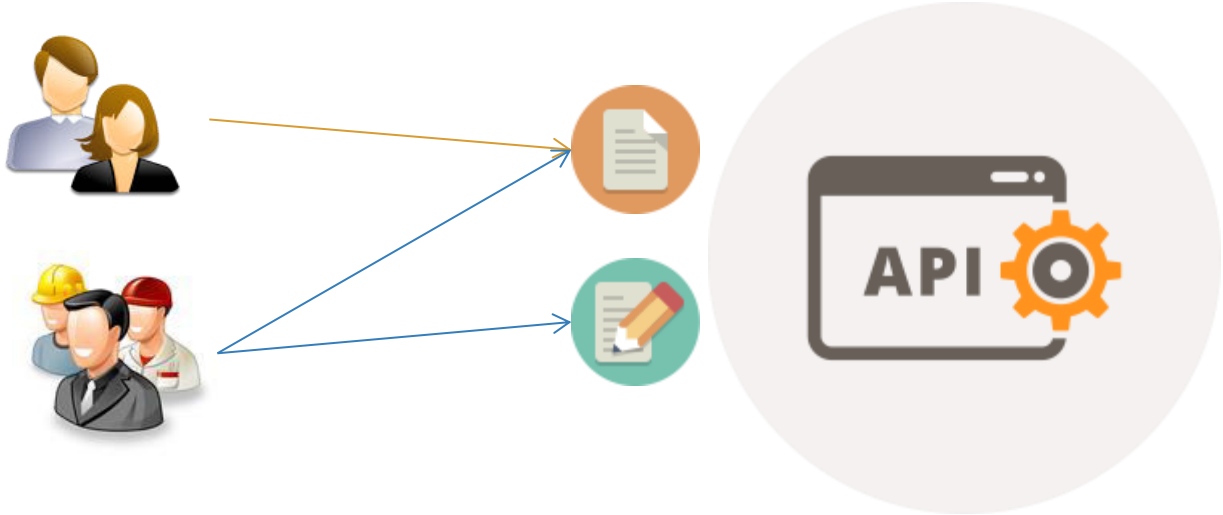
AUTORIZACIÓN EN SERVICIOS WEB CON SPRING

SEGURIDAD DE SERVICIOS RESTful

- El cliente puede acceder, pero, ¿a que datos?.

AUTORIZACIÓN CON RESTful

- Una vez que el cliente ha accedido, podemos definir diferentes niveles dentro de nuestro API, en base a ROLES.



AUTORIZACIÓN CON RESTful

- Mapeo de URLs
- Anotaciones de métodos con ROLES

AUTORIZACIÓN CON RESTful
Mapeo de URLs

```
@Override
protected void configure(HttpSecurity http) throws Exception {

    http.csrf().disable()
        .authorizeRequests()
        .antMatchers(HttpMethod.GET,"/api/hello").hasRole("USER")
        .antMatchers(HttpMethod.GET,"/api/admin/hello").hasRole("ADMIN")
        .antMatchers(HttpMethod.POST,"/api/admin/hello").hasRole("ADMIN")
        .and().httpBasic();

}
```

AUTORIZACIÓN CON RESTful
Anotación de métodos

```java
@RestController
@RequestMapping("api")
public class SampleController {

    @PreAuthorize("hasAnyRole('ADMIN','USER')")
    @GetMapping("/hello")
    public String sayHello() {
        return "Hello World!";
    }

    @PreAuthorize("hasRole('ADMIN')")
    @GetMapping("/admin/hello")
    public String sayHelloForAdmins() {
        return "Hello World, Mr. Admin";
    }

    @PreAuthorize("hasRole('ADMIN')")
    @PostMapping("/admin/hello")
    public String submitGreet(@RequestBody String greet) {
        return "The recieved greet is " + greet;
    }

}
```

```java
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfiguration extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {

        http.csrf().disable()
            .authorizeRequests()
            .anyRequest().authenticated()
            .and()
            .httpBasic();

    }
```