Sobre mí

- Luis Armando Ramírez Aguilar
- Cofundador, socio y líder técnico en SISU Technologies
- Desarrollo de aplicaciones web y móviles
- Java para el desarrollo de aplicaciones web: Spring
- Twitter: @cenobyte321
- E-mail: aramirez@sisu.mx

Spring Security OAuth 2.0

El problema a resolver

Los actores de OAuth

Resource Owner

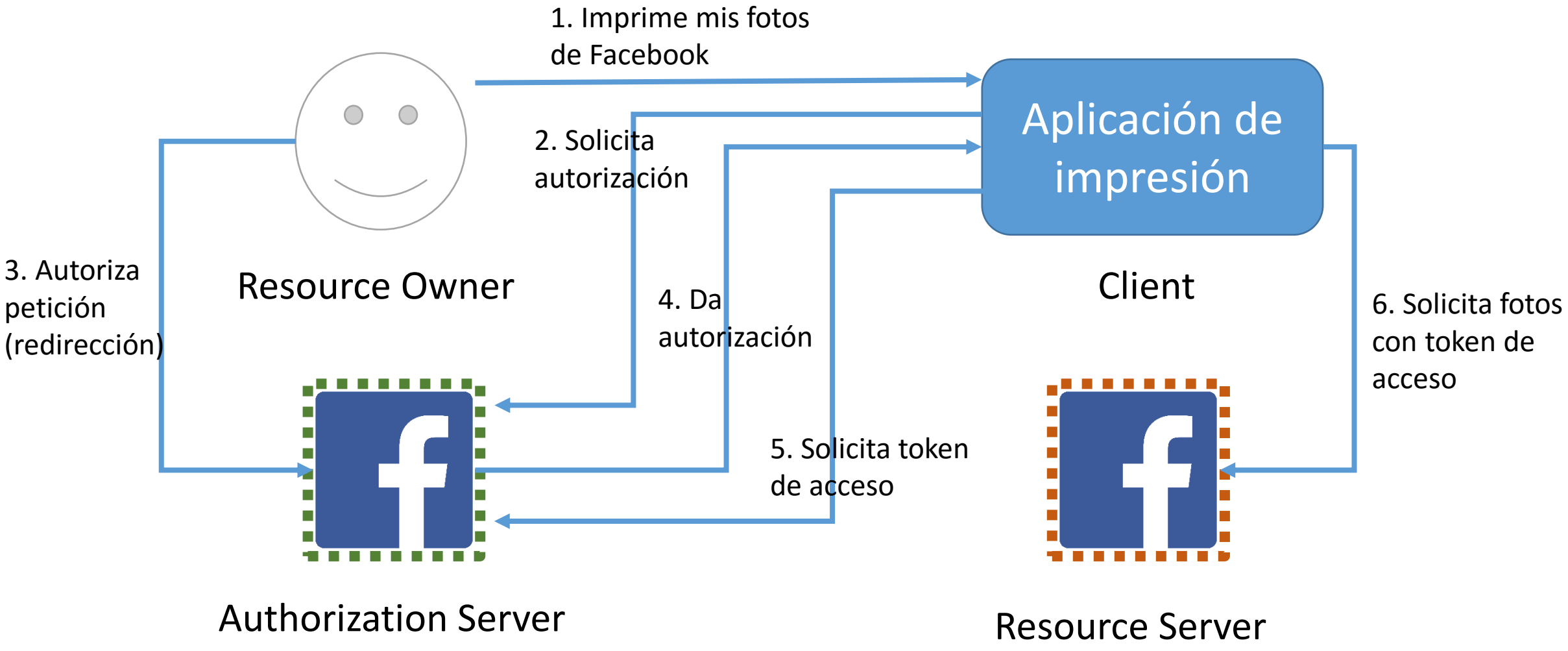
Aplicación de impresión

Client

Authorization Server

Resource Server

Flujo base


1. Imprime mis fotos de Facebook
2. Solicita autorización
3. Autoriza petición (redirección)
4. Da autorización
5. Solicita token de acceso
6. Solicita fotos con token de acceso
Aplicación de impresión
Client
Resource Owner
Authorization Server
Resource Server

Historia de OAuth

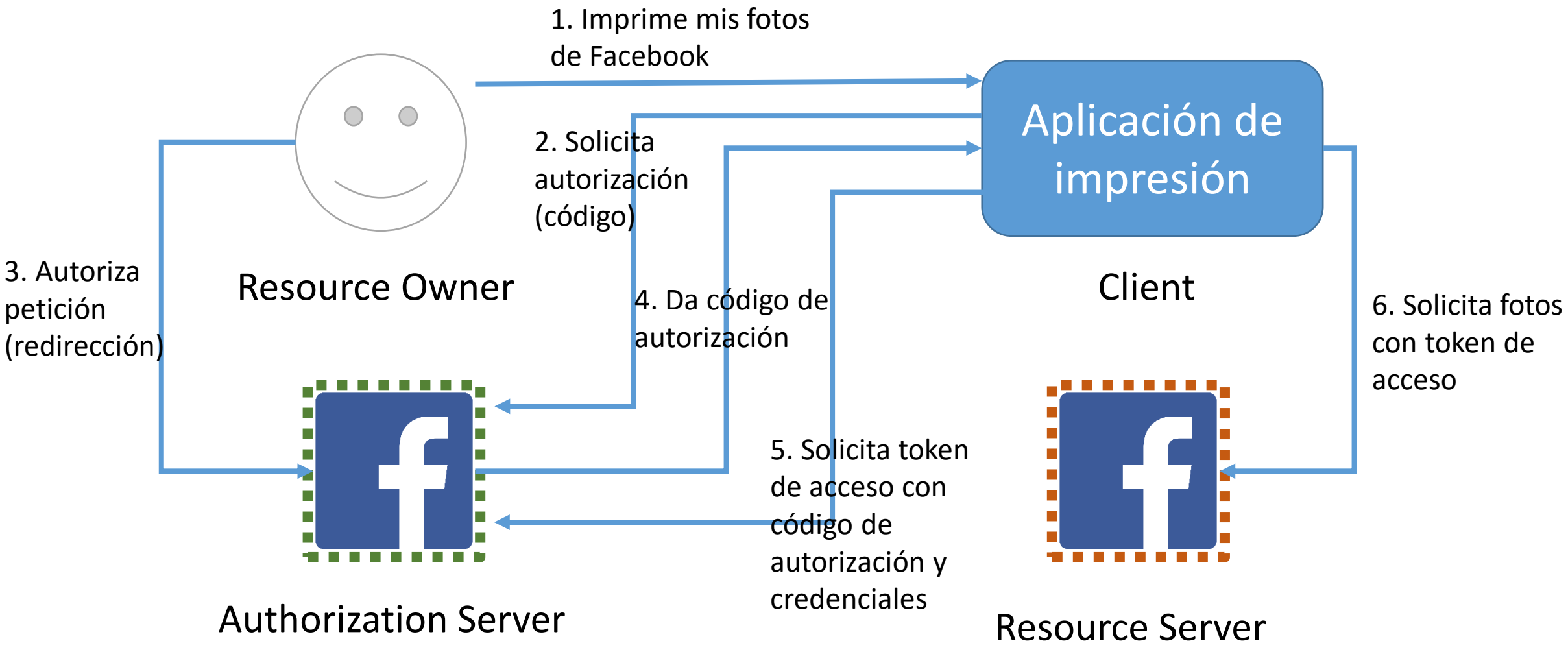
2012 OAuth 2.0 por el IETF – RFC 6749

- Más sencillo que la versión 1.0(a)
- Recae en HTTPS
- Pensado para aplicaciones de escritorio, móviles y web
- Marco de trabajo
- No es considerado como una mejora a OAuth 1.0a
- Controversial

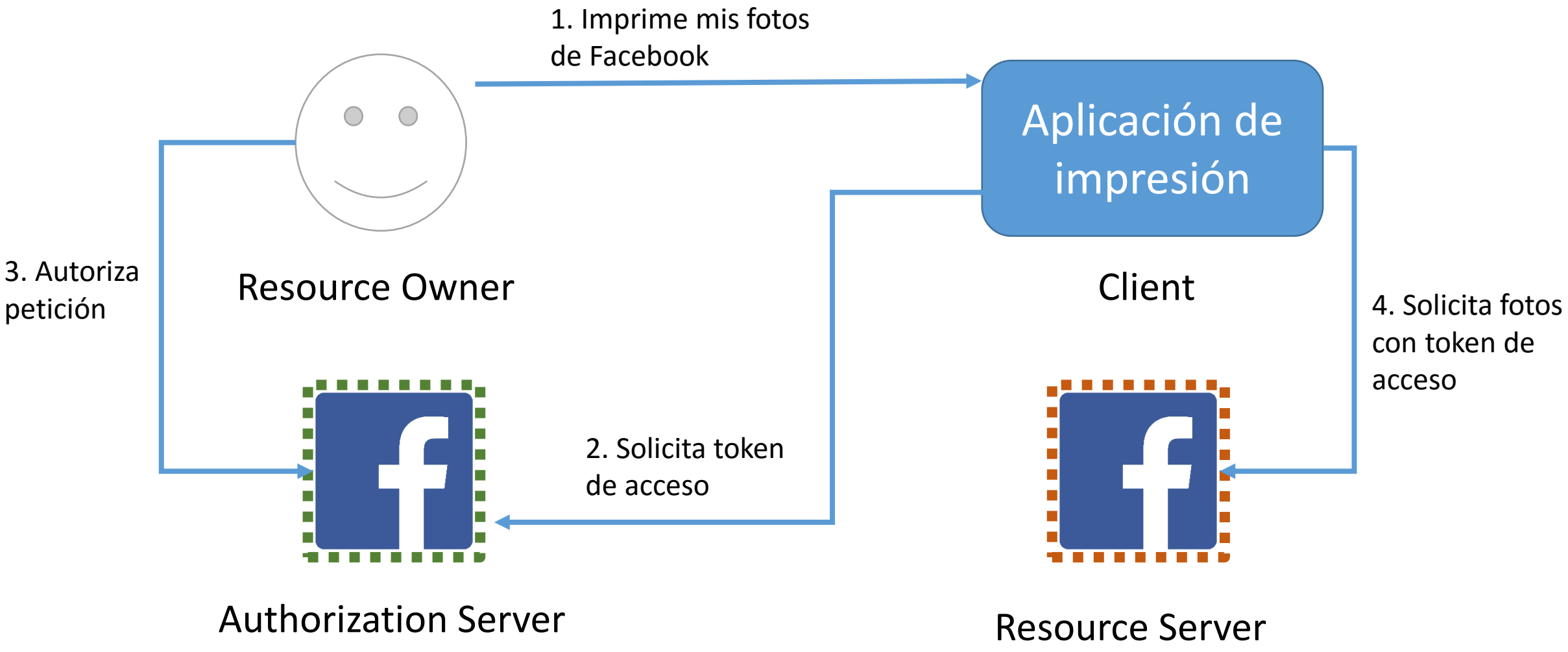
Flujos de OAuth 2.0

- Authorization code – Request - response
- Implicit - Aplicaciones web (no confiables)
- Resource Owner Password Credentials - Aplicaciones móviles y de escritorio (“confiables”).
- Client credentials - Clientes

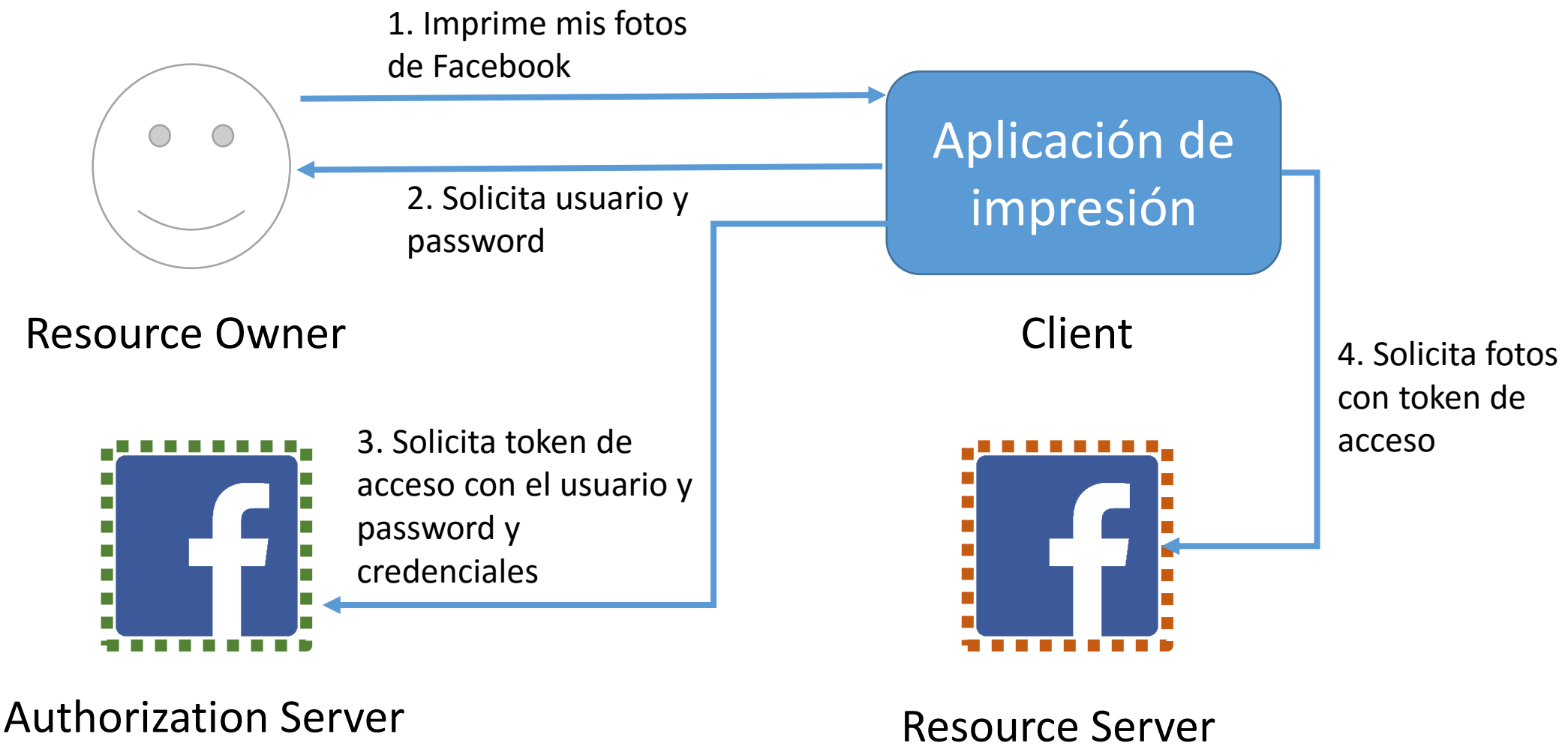
Authorization code



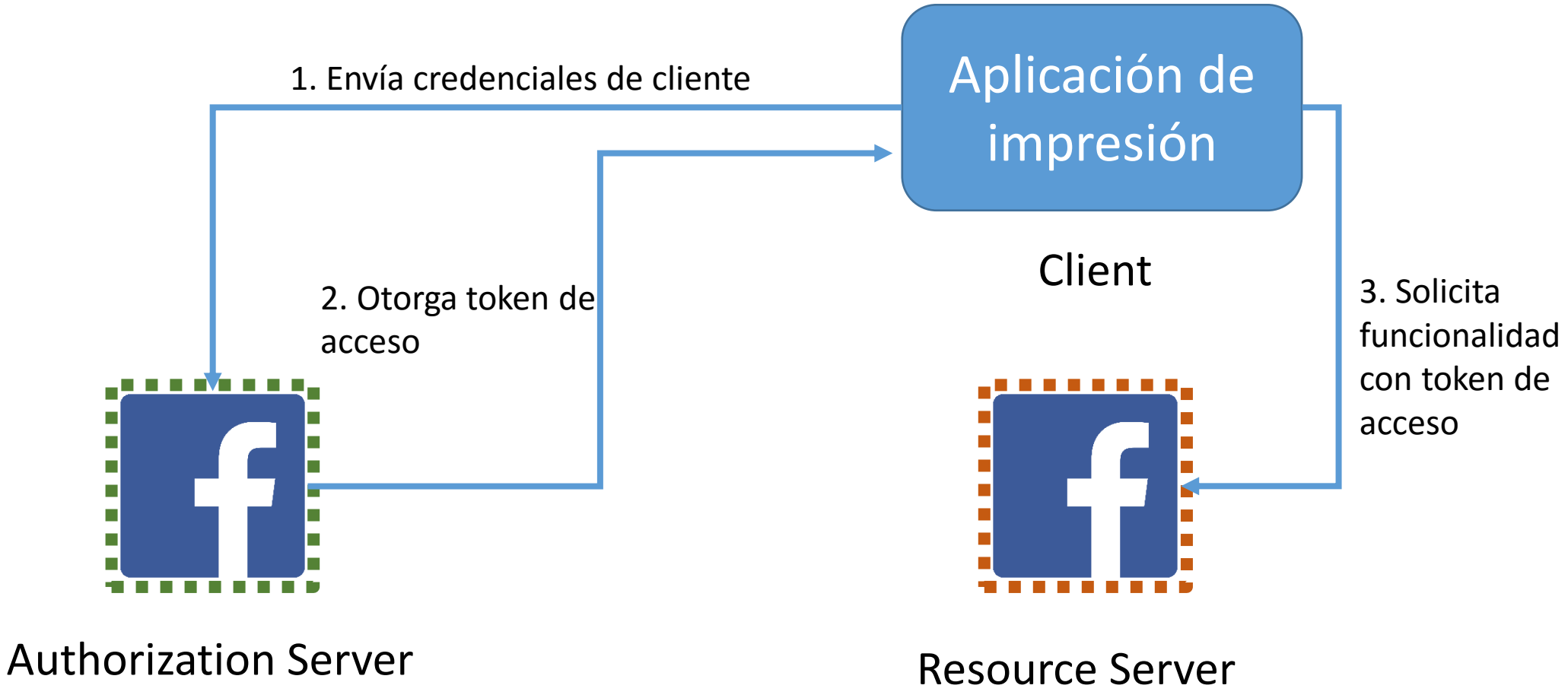
Implicit


1. Imprime mis fotos de Facebook
Aplicación de impresión
Client
Resource Owner
3. Autoriza petición
4. Solicita fotos con token de acceso
2. Solicita token de acceso
Authorization Server
Resource Server

Resource Owner credentials



Client credentials



Scopes

- Forma de dar una autorización más fina a los recursos
- Ejemplo: Discriminar el nivel de acceso de cada cliente
- Strings aleatorios:
- Ejemplo LinkedIn
  - r_fullprofile
  - w_share
  - r_emailaddress

Refresh tokens

- Los tokens de acceso deben de siempre tener un tiempo de expiración.
- Mecanismo para solicitar de nuevo un token de acceso a petición del cliente
- Incompatibles con los flujos: “implicit” y “client credentials”

Spring Security OAuth 2.0

- Sub-proyecto de Spring Security
- Tiene implementaciones para 1.0a y 2.0
- Configuración por XML o por medio de Java
- Sencillo de implementar*

Incluimos Spring Security OAuth 2.0 en nuestro proyecto

```
<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.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.security.oauth</groupId>
        <artifactId>spring-security-oauth2</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

Spring Security OAuth 2.0

Resource Owner

Aplicación de impresión

Client

Authorization Server

Resource Server

Spring Security OAuth 2.0

Resource Owner

Aplicación de impresión

Client

@EnableAuthorizationServer

@EnableResourceServer
@RestController, etc.

Ponemos algunos recursos

```java
@RestController
@RequestMapping(value="/api")
public class APIController {

    @Autowired
    TokenStore tokenStore;

    @RequestMapping(value="/saluda", method= RequestMethod.GET)
    public ResponseEntity<String> test(){
        return new ResponseEntity<String>("Hola mundo", HttpStatus.OK);
    }

    @RequestMapping(value="/user/posts")
    public ResponseEntity<List<PostDTO>> getPosts(){

        List<PostDTO> posts = new ArrayList<>();
        for(int i = 0; i < 10; i++){
            PostDTO post = new PostDTO();
            post.setAutor("Autor " + i);
            post.setCuerpo("Cuerpo " + i);
            post.setTitulo("Título " + i);
            post.setFechaPublicacion(new Timestamp(System.currentTimeMillis()));
            posts.add(post);
        }
        return new ResponseEntity<>(posts, HttpStatus.OK);
    }
```

Configuramos el servidor de recursos

```
@Configuration
@EnableResourceServer
public class ResourceServerConfiguration extends ResourceServerConfigurerAdapter{

    @Autowired
    TokenStore tokenStore;

    @Override
    public void configure(ResourceServerSecurityConfigurer resources) {
        resources
        .resourceId("resource")
        .tokenStore(tokenStore);
    }
    @Override
    public void configure(final HttpSecurity http) throws Exception {
        http.antMatcher("/api/**")
            .authorizeRequests().anyRequest()
            .access("#oauth2.hasScope('read')");
 }
}
```

Configuramos el servidor de autorización

```
@Configuration
@EnableAuthorizationServer
public class AuthorizationServerConfiguration extends AuthorizationServerConfigurerAdapter{

    @Autowired
    AuthenticationManager authenticationManager;

    @Override
    public void configure(AuthorizationServerEndpointsConfigurer endpoints) throws Exception {
        endpoints
        .authenticationManager(authenticationManager)
        .tokenStore(tokenStore())
        .tokenEnhancer(tokenEnhancer())
        ;
    }

    @Bean
    public TokenStore tokenStore() {
        return new InMemoryTokenStore();
    }
```

Configuramos el servidor de autorización

```
@Override
public void configure(ClientDetailsServiceConfigurer clients) throws Exception {
        clients
        .inMemory()
            .withClient("client_authorization_code")
            .secret("secret")
            .authorizedGrantTypes("authorization_code", "refresh_token")
            .scopes("read")
            .redirectUris("http://example.com", "http://localhost:8081/client/")
            .resourceIds("resource")
        .and()
            .withClient("client_implicit")
            .secret("secret")
            .authorizedGrantTypes("implicit")
            .scopes("read")
            .resourceIds("resource")
        .and()
            .withClient("client_password")
            .secret("secret")
            .authorizedGrantTypes("password", "refresh_token")
            .scopes("read", "write")
            .resourceIds("resource").accessTokenValiditySeconds(8000)
        .and()
            .withClient("client_client_credentials")
            .secret("secret")
            .authorizedGrantTypes("client_credentials")
            .scopes("other")
            .resourceIds("resource")
            ;
}
```

Ponemos algunos usuarios

```
@Configuration
@EnableWebSecurity
@Order(-11)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth
                .inMemoryAuthentication()
                    .withUser("myuser")
                    .password("mypassword")
                    .roles("USER")
                .and()
                    .withUser("test")
                    .password("testpassword")
                .roles("USER");
    }
```

Configuramos las formas de autentificación

```
@Configuration
@EnableWebSecurity
@Order(-11)
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {...}

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable();
        http.logout()
            .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
            .invalidateHttpSession(true)
            .logoutSuccessUrl("/login.html");

        http.authorizeRequests()
            .antMatchers("/login.html")
            .permitAll()

        .and()
            .formLogin()
            .loginPage("/login.html")
            .permitAll()
            .loginProcessingUrl("/login")
            .usernameParameter("username")
            .passwordParameter("password")
            .and()
            .requestMatchers().antMatchers("/login","/logout","/oauth/authorize")
            .and()
                .authorizeRequests().anyRequest().authenticated()
        .and().exceptionHandling().accessDeniedPage("/denied.html");

    }

}
```

Endpoints creados

Endpoint	Descripción
/oauth/token	Genera los tokens de acceso
/oauth/confirm_access	Pantalla para que el usuario autorice a la aplicación y sus respectivos “scopes”
/oauth/error	Muestra errores del servidor de autorización
/oauth/authorize	Genera los códigos de autorización (authorization code) y muestra los tokens (implicit)
/oauth/check_token	Verifica el estado del token*

```
@Override
public void configure(AuthorizationServerSecurityConfigurer oauthServer) throws Exception
{
    oauthServer.checkTokenAccess("isAuthenticated()");
}
```

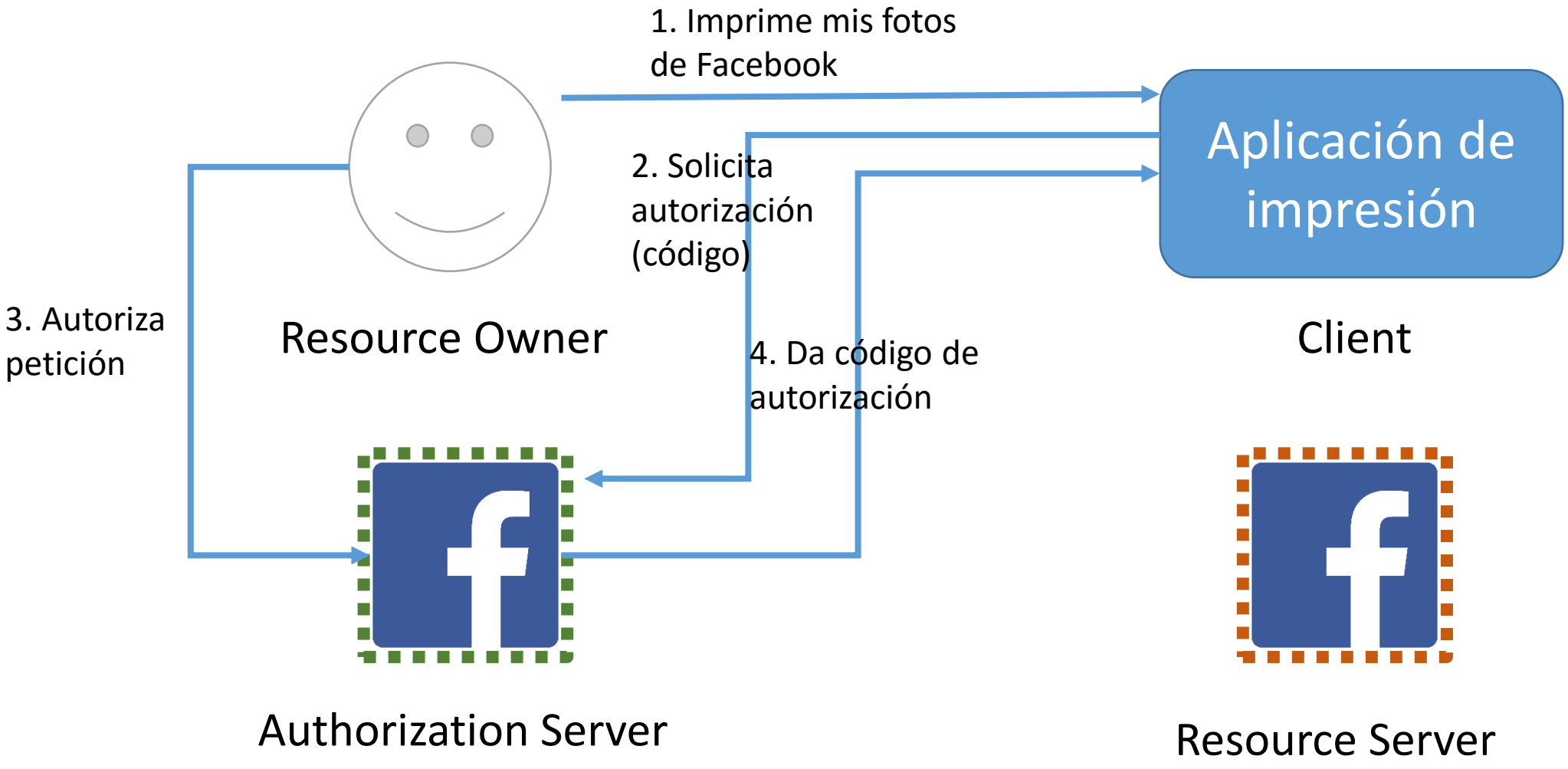
Authorization Code

1. Obtener el código de autorización, navegando a la URL:

http://localhost:8080/oauth/authorize?response_type=code&client_id=client_authorization_code&redirect_uri=http://example.com

- Parámetros:
  - response_type=code
  - client_id
  - redirect_uri

Authorization Code



Authorization Code

2. Obtener el token

curl -u client_authorization_code:secret -X POST http://localhost:8080/oauth/token -H "Accept: application/json" -d "grant_type=authorization_code&client_id=client_authorization_code&redirect_uri=http://example.com&code=ibs6g2"

- Autentificación del cliente
- Parámetros:
  - grant_type=authorization_code
  - client_id
  - redirect_uri
  - code

Authorization Code

```
sh-3.2$ curl -u client_authorization_code:secret -X POST http://localhost:8080/oauth/token -H "Accept: a
pplication/json" -d "grant_type=authorization_code&client_id=client_authorization_code&redirect_uri=http
://example.com&code=yQ4ybp"
```

```
{"access_token":"bd261781-8776-4753-acc3-60b1f381e4d8","token_type":"bearer","refresh_token":"dbf25302-7
ee7-40a4-87ef-58b42bff3963","expires_in":43131,"scope":"read","user_name":"myuser"}sh-3.2$
```

```
sh-3.2$ curl -u client_authorization_code:secret -X POST http://localhost:8080/oauth/token -H "Accept: a
pplication/json" -d "grant_type=authorization_code&client_id=client_authorization_code&redirect_uri=http
://example.com&code=yQ4ybp"
```

```
{"error":"invalid_grant","error_description":"Invalid authorization code: yQ4ybp"}sh-3.2$
```

Authorization Code

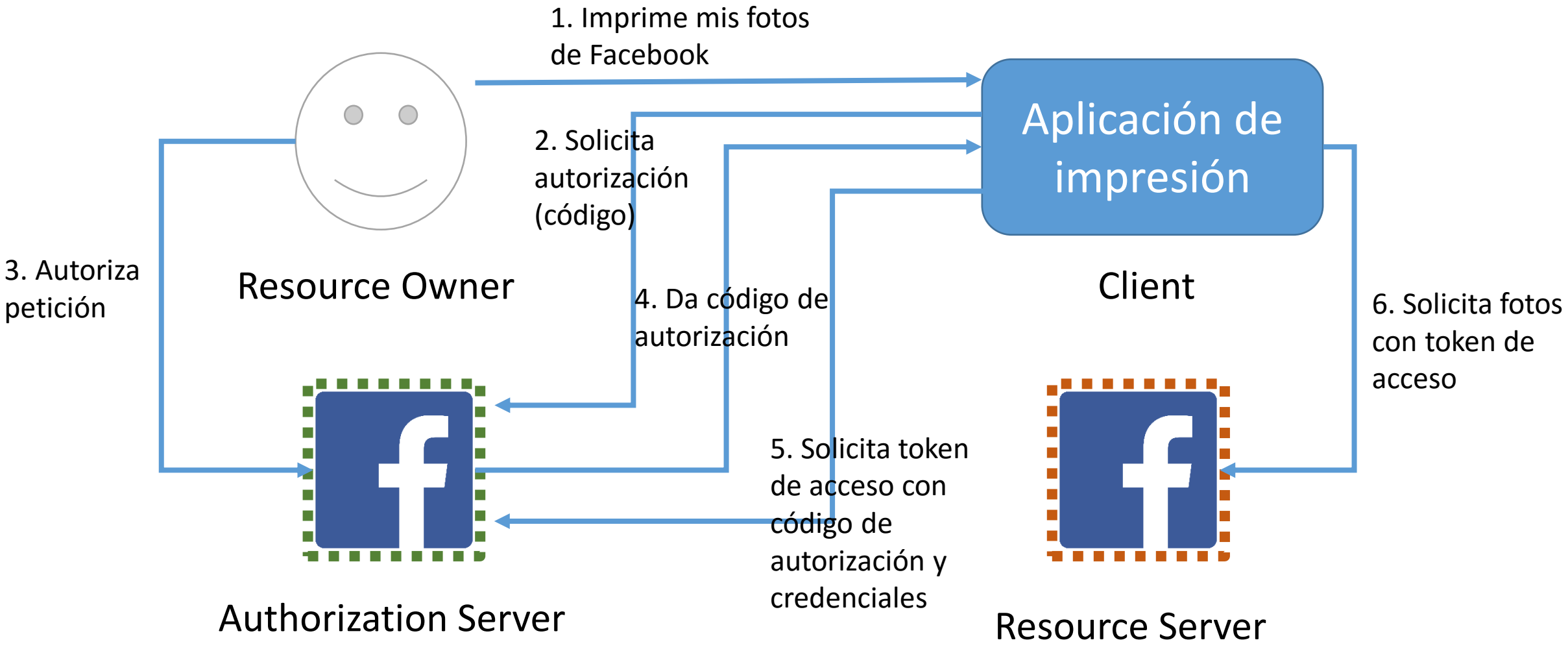
3. Obtener acceso al recurso

curl -X POST http://localhost:8080/api/user/posts -H "Authorization: Bearer 1b5e56de-5e79-4db1-bd58-a8aed6939667"

```
sh-3.2$ curl -X POST http://localhost:8080/api/user/posts -H "Authorization: Bearer bd261781-8776-4753-a
cc3-60b1f381e4d8"
```


```
[{"titulo":"Título 0","autor":"Autor 0","cuerpo":"Cuerpo 0","fechaPublicacion":1461190522527},{"titulo":
"Título 1","autor":"Autor 1","cuerpo":"Cuerpo 1","fechaPublicacion":1461190522527},{"titulo":"Título 2",
"autor":"Autor 2","cuerpo":"Cuerpo 2","fechaPublicacion":1461190522527},{"titulo":"Título 3","autor":"Au
tor 3","cuerpo":"Cuerpo 3","fechaPublicacion":1461190522527},{"titulo":"Título 4","autor":"Autor 4","cue
rpo":"Cuerpo 4","fechaPublicacion":1461190522527},{"titulo":"Título 5","autor":"Autor 5","cuerpo":"Cuerp
o 5","fechaPublicacion":1461190522527},{"titulo":"Título 6","autor":"Autor 6","cuerpo":"Cuerpo 6","fecha
Publicacion":1461190522527},{"titulo":"Título 7","autor":"Autor 7","cuerpo":"Cuerpo 7","fechaPublicacion
":1461190522527},{"titulo":"Título 8","autor":"Autor 8","cuerpo":"Cuerpo 8","fechaPublicacion":146119052
2527},{"titulo":"Título 9","autor":"Autor 9","cuerpo":"Cuerpo 9","fechaPublicacion":1461190522527}]sh-3.
```

Authorization Code



Cliente con Authorization Code

- RestTemplate con OAuth 2.0 incluido
- Simplifica el tener que programar el baile anterior
- Únicamente se configuran los parámetros del cliente

Cliente con Authorization Code

```
@Bean
protected OAuth2ProtectedResourceDetails resource() {
    AuthorizationCodeResourceDetails resource = new AuthorizationCodeResourceDetails();
    resource.setAccessTokenUri(tokenUrl);
    resource.setUserAuthorizationUri(authorizeUrl);
    resource.setClientSecret("secret");
    resource.setClientId("client_authorization_code");

    return resource ;
}
```

Cliente con Authorization Code

```
@Autowired
private OAuth2RestOperations restTemplate;

@RequestMapping("/")
public String home() {
    String result = restTemplate.getForObject(baseUrl + "/api/saluda", String.class);
    return result;
}

@Bean
public OAuth2RestOperations restTemplate(OAuth2ClientContext oauth2ClientContext) {
    return new OAuth2RestTemplate(resource(), oauth2ClientContext);
}
```

Implicit

1. Obtener el token de acceso, navegando a la URL:

http://localhost:8080/oauth/authorize?response_type=token&client_id=client_implicit&redirect_uri=http://example.com

- Parámetros:
  - response_type=token
  - client_id
  - redirect_uri

Implicit

- Ejemplo de respuesta:

http://example.com/#access_token=28605e78-b8d9-488a-a778-65dccb46134e&token_type=bearer&expires_in=43200&scope=read&user_name=myuser

- Respuesta:
  - access_token
  - token_type
  - expires_in
  - scope
  - user_name(?)

Token Enhancer

- Capacidad para agregar atributos adicionales junto con el token. Ejemplo: nombre de usuario

```
@Configuration
@EnableAuthorizationServer
public class AuthorizationServerConfiguration extends AuthorizationServerConfigurerAdapter{

    @Autowired
    AuthenticationManager authenticationManager;

    @Override
    public void configure(AuthorizationServerEndpointsConfigurer endpoints) throws Exception {
        endpoints
        .authenticationManager(authenticationManager)
        .tokenStore(tokenStore())
        .tokenEnhancer(tokenEnhancer())
        ;
    }
    @Bean
    public TokenEnhancer tokenEnhancer() {
        return new CustomTokenEnhancer();
    }
```

Token Enhancer

```
public class CustomTokenEnhancer implements TokenEnhancer {

    @Override
    public OAuth2AccessToken enhance(OAuth2AccessToken accessToken, OAuth2Authentication authentication) {
        User principal = (User)authentication.getPrincipal();
        Map<String, Object> additionalInfo = new HashMap<>();
        additionalInfo.put("user_name", principal.getUsername());

        ((DefaultOAuth2AccessToken) accessToken).setAdditionalInformation(additionalInfo);

        return accessToken;
    }

}
```

Resource Owner Password Credentials

Obtención del token

curl -u client_password:secret -X POST http://localhost:8080/oauth/token -H "Accept: application/json" -d "username=myuser&password=mypassword&grant_type=password"

- Autentificación del cliente
- Parámetros:
  - grant_type=password
  - username
  - password

Resource Owner Password Credentials

```
sh-3.2$ curl -u client_password:secret -X POST http://localhost:8080/oauth/token -H "Accept: application
/json" -d "username=myuser&password=mypassword&grant_type=password"
```


```
{"access_token":"55a70f6e-e8a2-47c4-a282-a8ed6f11ef35","token_type":"bearer","refresh_token":"ea6188ea-a
578-4c8b-acc8-d40063364624","expires_in":7994,"scope":"read write","user_name":"myuser"}sh-3.2$
```

Client Credentials

- Obtención del token

curl -u client_client_credentials:secret -d "grant_type=client_credentials" http://localhost:8080/oauth/token

- Autentificación del cliente
- Parámetros:
  - grant_type=client_credentials

- Hay un problema con nuestra configuración actual ¿por qué? (puntos extra)

Client Credentials

```
sh-3.2$ curl -u client_client_credentials:secret -d "grant_type=client_credentials" http://localhost:808
0/oauth/token
```

```
{"access_token":"09ab8e91-4134-46b5-a7d7-7a3314ca1a7e","token_type":"bearer","expires_in":42843,"scope":
"read"}sh-3.2$
```

Refrescar token

- Refrescar el token

curl -u client_authorization_code:secret -d "grant_type=refresh_token&client_id=client_authorization_code&refresh_token=a87416a6-4595-43a0-aab0-d55135165016" http://localhost:8080/oauth/token

- Autentificación del cliente
- Parámetros:
  - grant_type=refresh_token
  - client_id
  - refresh_token

Refrescar token

```
sh-3.2$ curl -u client_password:secret -d "grant_type=refresh_token&client_id=client_password&refresh_to
ken=ea6188ea-a578-4c8b-acc8-d40063364624" http://localhost:8080/oauth/token
```

```
{"access_token":"48df6e6b-d2b6-4b22-842b-947dd2197312","token_type":"bearer","refresh_token":"ea6188ea-a
578-4c8b-acc8-d40063364624","expires_in":7999,"scope":"read write","user_name":"myuser"}sh-3.2$
```

Verificación de token

Verificación del token:

curl -u client_password:secret -X POST http://localhost:8080/oauth/check_token -H "Accept: application/json" -d "token=bb900fb1-d1ad-460a-81f2-35ea7e621a1b"

- Autentificación del cliente
- Parámetros:
  - token

Verificación de token

```
sh-3.2$ curl -u client_password:secret -X POST http://localhost:8080/oauth/check_token -H "Accept: appli
cation/json" -d "token=55a70f6e-e8a2-47c4-a282-a8ed6f11ef35"
```

```
{"aud":["resource"],"exp":1461200400,"user_name":"myuser","authorities":["ROLE_USER"],"client_id":"clien
t_password","scope":["read","write"]}sh-3.2$
```

Revocar token

```
@RestController
public class OAuthExtra {

    @Autowired
    private DefaultTokenServices defaultTokenServices;

    @RequestMapping(value = "/oauth/revoke-token", method = RequestMethod.POST)
    @ResponseStatus(HttpStatus.OK)
    public void logout(HttpServletRequest request) {
        String authHeader = request.getHeader("Authorization");
        if (authHeader != null) {
            String tokenValue = authHeader.replace("Bearer", "").trim();
            defaultTokenServices.revokeToken(tokenValue);
        }
    }
}
```

Revocar token

Revocar

curl -X POST http://localhost:8080/oauth/revoke-token -H "Authorization: Bearer bb900fb1-d1ad-460a-81f2-35ea7e621a1b"

Obtener información del usuario

```
@RequestMapping(value="/user/getId")
public ResponseEntity<String> getId(){
    Authentication auth = SecurityContextHolder.getContext().getAuthentication();
    String clientName = ((OAuth2Authentication) auth).getUserAuthentication().getName();
    return new ResponseEntity<String>(clientName, HttpStatus.OK);
}
```

curl -X POST http://localhost:8080/api/user/getId -H "Authorization: Bearer 1b5e56de-5e79-4db1-bd58-a8aed6939667"

Conclusiones

- OAuth 2.0 es un marco de trabajo con mayor uso (LinkedIn, Facebook, Google, Twitter). Por lo que es necesario comprender sus flujos.

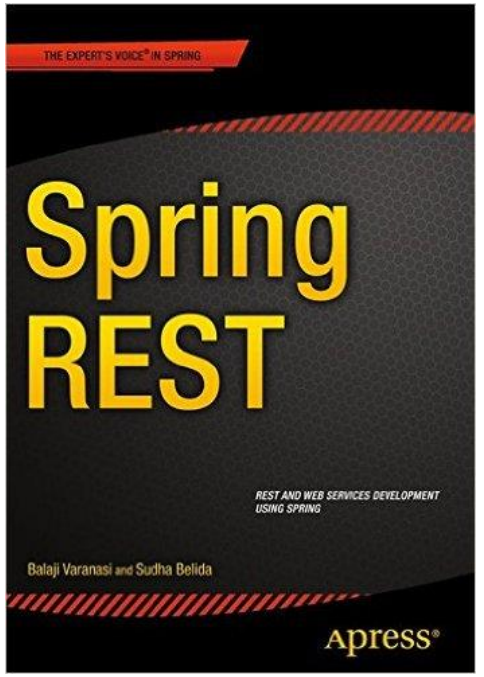
- Spring Security OAuth 2.0 da una forma sencilla para implementar un mecanismo de autorización. Spring Security se queda con la tarea de autentificar.

- Desventaja: Falta de documentación a fondo.

Fuentes de información

- "An Introduction to OAuth 2." *An Introduction to OAuth 2 | DigitalOcean*. Digital Ocean. Web. <https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2>.

- Varanasi, Balaji, and Sudha Belida. *Spring REST*. Berkeley, CA: Apress, 2015. Print.



Fuentes de información

- "Spring Security and Angular JS." Spring. Web. <https://spring.io/guides/tutorials/spring-security-and-angular-js/>.

- "OAuth 2 Developers Guide." Spring Security OAuth. Web. <http://projects.spring.io/spring-security-oauth/docs/oauth2.html>.

- "Spring-projects/spring-security-oauth." GitHub. Web. <https://github.com/spring-projects/spring-security-oauth/tree/ac142a6ce0c948aeb07837f7e8704dad4d3e3134/tests/annotation>.

Fuentes de información

- "OAuth Core 1.0 Revision A." *OAuth Core 1.0a*. 24 June 2009. Web <http://oauth.net/core/1.0a/>.

- "OAuth 2.0." *OAuth 2.0*. Web. <http://oauth.net/2/>.

- "Spring-projects/spring-security-oauth - SQL Schema." *GitHub*. Web. <https://github.com/spring-projects/spring-security-oauth/blob/master/spring-security-oauth2/src/test/resources/schema.sql>.

Sobre mí

- Luis Armando Ramírez Aguilar
- Cofundador, socio y líder técnico en SISU Technologies
- Desarrollo de aplicaciones web y móviles
- Java para el desarrollo de aplicaciones web: Spring
- Twitter: @cenobyte321
- E-mail: aramirez@sisu.mx

Imágenes

- Facebook logo: https://upload.wikimedia.org/wikipedia/commons/thumb/f/fb/Facebook_icon_2013.svg/2000px-Facebook_icon_2013.svg.png
- Twitter logo: http://eledelengua.com/imagenes/twitter-logo.png
- OpenID logo: https://openid.net/wordpress-content/uploads/2014/09/openid-r-logo-900x360.png
- Oauth logo: http://dret.net/lectures/mobapp-spring10/img/oauth.png
- Phishing: http://www.revistaproware.com/wp-content/uploads/2013/07/phishing.png
- SIGABA: https://upload.wikimedia.org/wikipedia/commons/f/fb/SIGABA-patent.png
- Facebook Login: http://i.stack.imgur.com/Wc2T1.png
- Session fixation : https://hueniverse.com/2009/04/23/explaining-the-oauth-session-fixation-attack/
- Valet parking: http://3.bp.blogspot.com/_lBcY8Pp43Ew/TBrKK9y_2KI/AAAAAAAAA_4/QOSmdEGg7WA/s200/valet-park.jpg